VST
PRELIMINARY DESIGN REVIEW

SECTION 6

SOFTWARE CONTROL SYSTEM
10/01/1999

Designed by :	**D. Mancini, M. Brescia**
Signature	[illegible] [illegible]
Prepared by :	**M. Brescia**
Signature	[illegible]
Supervised by :	**D. Mancini**
Signature	[illegible]
Verified by :	**D. Mancini, G. Sedmak, V. Fiume Garelli**
Signature	[illegible] [illegible] [illegible]
Approved by :	**G. Sedmak**
Signature	[illegible]

EVOLUTION RECORD

Issue	Revision	Date	Notes
1.0	0	01/12/98	First release for comments
1.1	1	10/12/98	Revised document organization
1.2	2	22/12/98	Some distributed corrections

TABLE OF CONTENTS

TABLE & FIGURES INDEX

6. SOFTWARE CONTROL SYSTEM

6.1 INTRODUCTION

6.1.1 Purpose

This document contains a set of baselines in both design and development of VST Telescope Control software. It takes into account the specifications and references underlined in the VLT Common software documentation, represents the software section for the Preliminary Design Review (PDR) document related to the VST project, and contains main functional specifications for VS Telescope software design and development where the main requirements come from [16]. The document is intended written for software and hardware development groups in order to explain all main guidelines for software realization. The section 5.2 is a brief overview of VLT Common Software baselines, particularly referred to TCS architecture. The other sections go fully into the VST Control Software description.

6.1.2 Scope

The VST control software is based on object oriented design principles, and interfaces with ESO VLT Common Software.

6.1.3 VST Software Documentation Plan

Along the VST project a complete package of software design documentation will be organized by VST software group. It will be composed by following documents:

- VST TCS - Functional Specification
- VST TCS - System Architecture Description
- VST Integration User Manual
- VST Optics Control LCU User/Maintenance Development module
- VST Altitude and Azimuth LCU User/Maintenance Development module
- VST Rotator LCU User/Maintenance Development module

6.1.4 Reference Documents

[1] VLT-MAN-ESO-17200-0888, VLT Common software - Overview, Issue 1.0, 17.08.95
[2] VLT-MAN-ESO-17230-1541, TCS Integration Module - tcsBuild User Manual, Issue 1.0, 08.01.98
[3] VLT-SPE-ESO-10000-0200, VLT Programme TCS Requirements Specification, Issue 1, 29.06.92
[4] VLT-SPE-ESO-11720-0001, VLT Software TCS Functional specification, Issue 1.1, 02.03.95
[5] VLT-ICD-ESO-17230-0641, Interface Control Document, Issue 1.0, 08.12.94
[6] VLT-MAN-ESO-17210-1369, VLT Software CCS-LCU NET01 User manual, Issue 1.0, 06.06.97
[7] VLT-SPE-ESO-17230-0819, TCS Mode Switching Design Description, Issue 1.0, 05.05.95
[8] VLT-SPE-ESO-17230-0797, VLT Software TCS preset Design Description, Issue 1.0, 19.04.95
[9] VLT-SPE-ESO-17230-0941, TCS Interface Design Description, Issue 1.0, 17.11.95
[10] VLT-SPE-ESO-17230-0755, TCS Tracking Design Description, Issue 1.0, 16.05.95
[11] VLT-MAN-ESO-17210-0431, CCS-LCU Time Interface Module User Manual, Issue 1.4, 09.02.95
[12] VLT-VER-ESO-17210-0433, Time interface Module Driver Test Procedure, Issue 1.3, 06.10.94
[13] VLT-PLA-ESO-00000-0007, VLT Software Test Plan, draft/3, 20.12.91
[14] VLT-SPE-ESO-17230-0941, TCS Interface Design Description, Issue 1.0, 17.11.95
[15] VLT-SPE-ESO-17230-0797, TCS Preset Design Description, Issue 1.0, 19.04.95
[16] VST Technical Proposal, TWG Naples, 06.10.97
[17] VLT-MAN-SBI-17210-0001, LCU Common Software User Manual, Issue 3.4, 05.05.97

[18] VLT-MAN-ESO-17210-0375, CCS-LCU Driver Development Guide, Issue 2.2, 11.06.96
[19] VLT-MAN-ESO-17210-0667, Guidelines for development of VLT App. SW, Issue 1.0, 3.12.96
[20] VLT-INS-ESO-00000-0573, Guide for preparation of Sw Design Description, Issue 1.0/1, 21.1.94
[21] VLT-INS-ESO-00000-0570, Guide for preparation of VLT Sw Documents, Issue 1.0/1, 21.01.94
[22] VLT-SPE-ESO-11320-0763, Alt and Az LCU control Sw Func Spec, Issue 1.0, 19.12.94
[23] VLT-SPE-ESO-17130-1210, LCU axis ctrl Module Design description, Issue 1.0, 20.10.97
[24] VLT-PRO-ESO-10000-0228, VLT Sw Programming Standards, Issue 1.0, 10.03.93
[25] Hexapod positioning system, Operating Manual MS 54E, Physik Instrumente, Release 1.1
[26] OmegaCAM: A wide field camera for the VST, Proposal to ESO, October 6, 1998, Draft

6.1.5 Abbreviations and Acronyms

ALT	ALTitude
AZ	AZimuth
CCD	Charge Coupled Device
CCS	Central Common Software
DEC	DECIination
DFS	Data Flow System
ESO	European Southern Observatory
EUI	Engineering User Interface
GUI	Graphical User Interface
HBS	Hydrostatic Bearing System
HOS	High Level Operation Software
INS	Instrumentation Software
LCC	LCU Common Software
LCU	Local Control Unit
MSM	Mechanical Software Module
OAC	Osservatorio Astronomico di Capodimonte
PDR	Preliminary Design Review
PHA	Preliminary Hazard Analysis
PI	Physik Instrumente
RA	Right Ascension
ROS	Remote Operation Software
RTAP	Real-Time Application Platform
SCPI	Standard Command Programmable Interface
SH	Shack Hartmann
SSHA	SubSystem Hazard Analysis
TBC	To Be Confirmed
TBD	To Be Defined
TCS	Telescope Control Software
TIM	Time Interface Module
TRS	Time Reference System
TWG	Technology Working Group
UIF	(Portable) User Interface Toolkit
UTC	Universal Time Coordinated
VLT	Very Large Telescope
VME	Versa Module Eurocard
VST	VLT Survey Telescope
WS	WorkStation

6.2 VLT Common Software

The VLT Common Software is a set of basic tools useful for developing applications on workstations and/or LCUs. Some of main applications are:
Utilities for the application of standards to code and document development;
LCU Common Software (LCC);
Drivers for the standard LCU boards;
The motion control;
The Workstation Common Software (CCS);

As for any software development, for the VLT Common Software it is possible to identify:
The process (how to develop the software);
The product (what will be output at the end of the process);

The process consists of:
Software life cycle (it defines what, documents and code, and when shall be provided);
Standards and tools to be used for the development;
The way of keeping control of the product;
The way of checking the product respect both to the functional and process requirements;

The products for the VLT Common Software have been divided into:
CCS, Central Control Software;
HOS, High Level Operations Software;
INS, Instrumentation Software;
ROS, Remote Operation Software;
TCS, Telescope Control Software;

The VLT Software Programming Standards provides a set of rules to be followed during the design and development of VST control software to be used in conjunction with VLT Software. It contains instructions for the organization of a software module, naming conventions, C language, script files etc. For the software testing phase, there is also a standard set of rules that will be followed in VST software, based on three levels:

- Module;
- Integration;
- Acceptance;
-

These three rules are fully described in [13]. The main idea is to standardize the way of preparing and invoking a test in order to allow automatic testing and to reduce test Documentation. As main rule, each software module should be independently testable.

6.2.1 CCS and LCC Software Description

The core of the VLT Common Software is the Central Control Software (CCS) and its VxWorks implementation, the LCU Control Software (LCC). Their behavior is practically the same in terms of main services and interfaces, but with different implementation. Of course, they also provide a set of specific services. The CCS is built on top of RTAP that provides the basic services for real time database and interprocess communication. The mentioned services provided by the CCS and LCC are:

<u>services for both WS and LCU:</u>
general (ccs);
logging (log);
event handling (evt);
error system (err);
message system (msg);
database (db);
time handling (tims);
alarm management (alrm);
access control (book);
command handling (cmd);
common access interface (cai)

<u>services for interaction between WS and LCU:</u>
Time synchronization system (ntp);
LCU database variables monitoring (scan);
Database variable sampling tool (samp);

<u>services only for LCU:</u>
LCU management;
I/O Signal Handling;
LCU access Control;
Command interpreter, that provides a standard architecture for LCU applications;
A template application using the Command Interpreter (citmp);

<u>services only for WS:</u>
Extended CCS, CCS C++ interface (eccs);
Event handler (evh, evhEt, fnd) that provides a standard architecture for WS applications;

<u>software tools:</u>
Graphical User Interface (GUI) development kit, that allows to build, on a WS, graphical panels able to access local and remote database and to interact, through the message system, with WS and/or LCU applications. It is a powerful tool to build both target UIF and test or debugging utilities for both WS and LCU development.
UIF Panel Editor (CCS/Panel);
UIF Widget library (CCS/uif);
WS and LCU database development tools, like Database Loader (CCS/dbl), Database Class Browser (CCS/dblcb);
WS and LCU Debugging tools, like CCS Engineering Interface (ccsei) that provides a set of graphical tools to perform typical debugging and test activities like exchanging commands to applications, monitoring database or logs, etc.;

6.2.2 WS-LCU Communication Software

The communication software consists of two modules:

Lqs (LCU), that provides the communication between the LCU and the outside world;
Qsemu (WS), that, if RTAP is not available, provides a communication engine for the WS to support a subset of the Message System;

6.2.3 HOS/Sequencer

The Sequencer is a general purpose tool, containing also TCL/TK, which allows execution of a predefined series of commands and development of no-time critical or test applications.

6.2.4 Drivers

Drivers form a family of software modules interfacing the hardware boards directly. These modules are:

Driver log utility (lculog), common utility for all drivers;
Driver common utilities and graphical interactive debug utility (lcudrv), common utility for all drivers;
Analog I/O driver (aio);
ACROMAG Digital I/O driver (acro);
Time Board driver (tim);
MACCON driver (mcon);
Servo Amplifier driver (ampl);
Heidenhain IK320 encoder (ikon);
Engineering User Interfaces for direct access to the driver callable interface: there is one panel for each standard board;

Drivers should not be accessed directly by applications, but via higher level software (LCC, Motor Control). In case of direct access, it is recommended the interaction with ESO.

6.2.5 Motor Control Module

The Motor Control Module (MCM) forms a service layer between application level and device driver level to provide a standardized, hardware independent interface between applications and motors. An application attaches a required motor and performs operations with that motor by use of Motor Control Module routines without knowing details about drivers or boards. The MCM is designed for a standard LCU environment, consisting of a VMEbus system with a CPU board, running the VxWorks operating system, version 5.1 or higher. The MCM makes extensive use of the LCU Common Software facilities, such as the Event Logging, Error Handling and Signal Handling facilities and the Local Database module.

6.3 VST Telescope Software System

6.3.1 *General Requirements*

The VLT Common Software and ESO TCS is quite applicable to VST project. The exact release will be the latest one available at the VLT-Paranal at the commissioning time of the VST telescope. This, of course, implies the use of UNIX and VxWorks (TORNADO) operative environments for WS and LCU respectively, as defined in the VLT standards. In all VST project, whenever possible will be adopted the ESO philosophy and software solutions.
In order to implement a VST software module, the following ESO-like approach will be considered:

each software module will be identified by a name and implemented as a directory structure;
any source file will be in the *src/* subdirectory;
an empty template for the source file will be get;
the code will be inserted in the source file;
a makefile will be created in the same subdirectory;
finally the executable software will be generated;

From the VST software development point of view, the software and the documents referred to VLT Common Software will be needed for:

install standardized development tools like compiler, make, etc;
develop WS applications using the services and modules provided by the CCS;
develop LCU applications using the services and modules provided by the LCC and Motor control, including the handling of LCU standardized boards;

6.3.1.1 Hardware Configuration

The VST hardware and software configuration in terms of CPUs and control system will follow the standards proposed by ESO.

6.3.1.1.1 TCS Workstation Configuration

For the TCS workstation it is foreseen the use of HP 9000 series. Of course it will be taken into account any upgrading during project development.

6.3.1.1.2 LCU Hardware Configuration

For the LCUs foreseen in the VST control system, the following hardware devices will be provided:

Motorola MVME 167 or Power PC CPU board - any version (taking into account the new release of ESO software).
Vmic VMIVME-311 Analog I/O board - version ID 0x0d
Acromag AVME948x Digital I/O board - any version
ESO Time Reference board - ESO internal version
Heidenhain IK320 V Encoder interpolation cards
MACCON VME4ST stepper motor controller and driver cards

The number of these devices for each LCU will depend from the single LCU use and it is fully described in the control hardware section of VST PDR document.

6.3.1.2 Software Configuration

In order to run VLT Common Software and integrate specific VST TCS application software in the standard hardware ESO environment described above, the following software will be used.

6.3.1.2.1 Workstation Software Configuration

HP-UX B.10.20, HP version of widely used UNIX operating System, latest release on which ESO has standardized.
Graphical libraries X11R5 and Motif 1.2.
RTAP 6.7, modular UNIX-based software platform for real-time data acquisition and control of continuous processes. In the VST the RTAP will be used to manage the database structure and database values on HP WS. The WS software is on top of RTAP and CCS isolates specific application software from UNIX-RTAP.
Tcl 7.6/Tk 4.2, programming system with very useful graphical interface facilities.
GNU utilities, used for program compilation provided by the Free Software Foundation.
ESO CCS and VLT Common Software modules, (latest release).

6.3.1.2.2 Lcu Software Configuration

TORNADO VxWorks operative system, development and execution environment for complex real-time and embedded applications. It will be the environment in which time critical VST telescope control software applications will run.
GNU utilities, used for program compilation provided by the Free Software Foundation.
ESO LCC and VLT Common Software modules, (latest release).

6.3.2 Telescope Control system Overview

At level of ESO HW standardization, the VST telescope control architecture is shown in Fig. 5.1. It consists of two workstations, respectively, for TCS and Guide/Acq. high level management and a series of LCUs, each dedicated to specific system control. All these processing units are connected together via Local Area Networks (LAN).

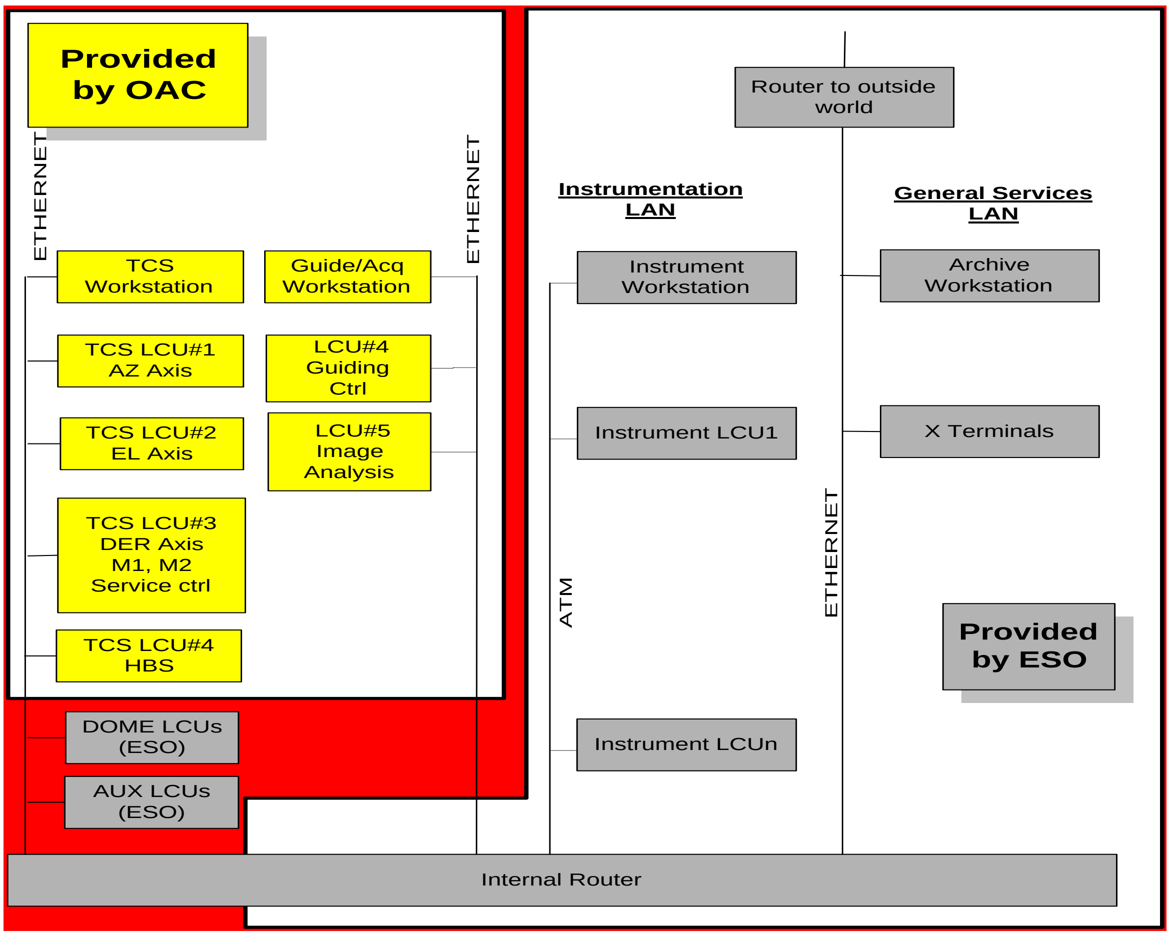


Fig. 5.1 - VST Main Control System Components Layout

The lowest level interface is obtained at level of LCUs. The task repartition is described below:

- TCS LCU#1 - Encoder system control for AZ axis
- TCS LCU#1 - AZ axis control (position loop)
- TCS LCU#1 - Control of other functions through specialized embedded controllers management

- TCS LCU#2 - Encoder system control for ALT axis
- TCS LCU#2 - EL axis control (position loop)
- TCS LCU#2 - Control of other functions through specialized embedded controllers management

- TCS LCU#3 - DER axis control
- TCS LCU#3 - M1 control
- TCS LCU#3 - M2 control
- TCS LCU#3 - Telescope auxiliary systems control through specialized embedded controllers management

- LCU#4 - Telescope Autoguiding system control
- LCU#4 - Technical CCD Camera Head and Array Control Electronics (ACE) management

- LCU#5 - Shack-Hartmann Image Analysis system control
- LCU#5 - Technical CCD Camera Head and Array Control Electronics (ACE) management

In the specialized embedded controlled functions are included the following tasks:

- Axes low level control
- Speed loop
- Temperature monitoring
- Local cooling control
- Motor interlocks
- Thermal control

6.3.3 Telescope Control Software

6.3.3.1 General Overview

The Telescope Control Software (TCS) is the software package that performs the following tasks:
Controls and monitors the telescope
Interfaces to users and other VST packages
Interfaces to and gets data from star catalogues

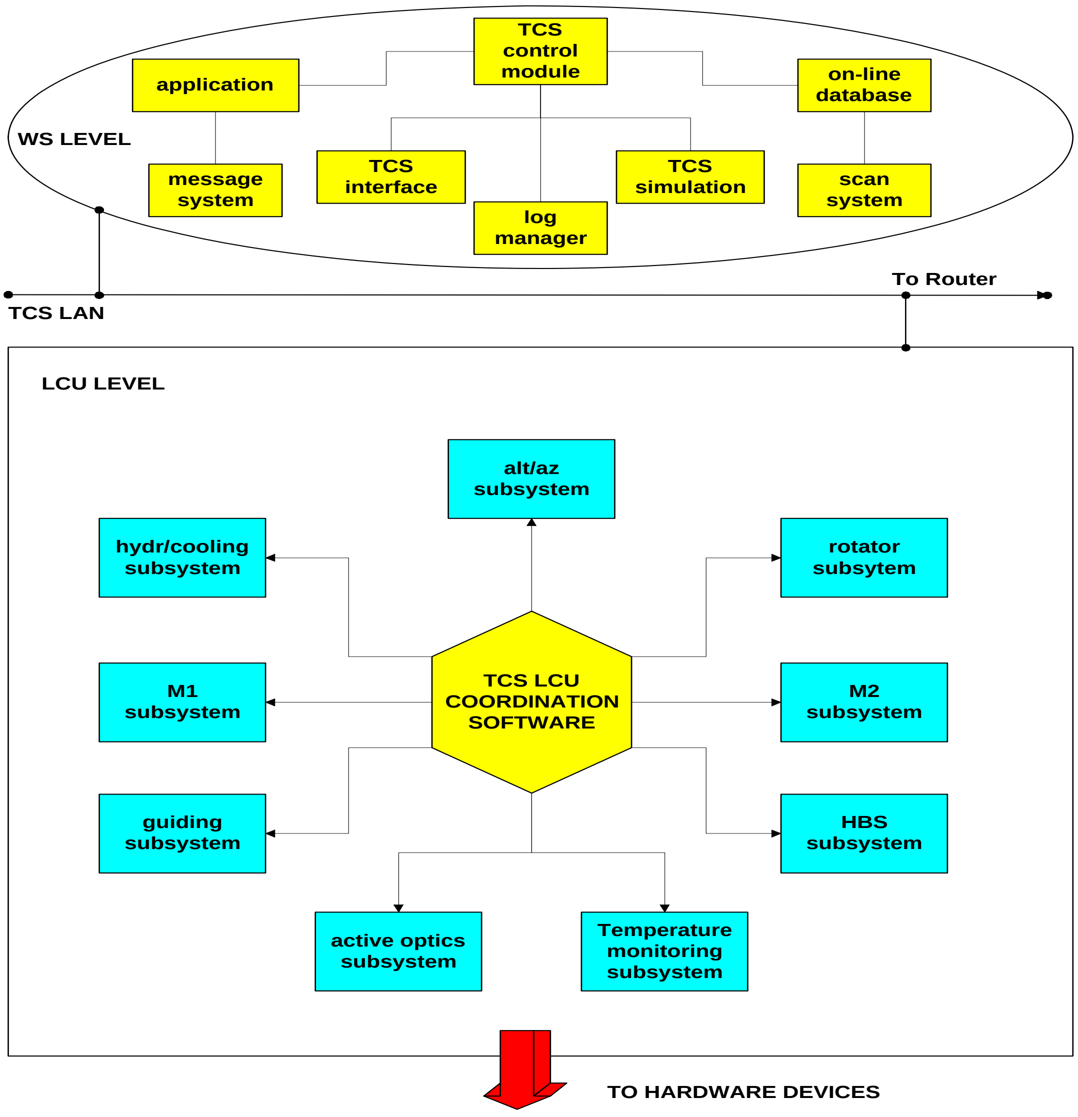


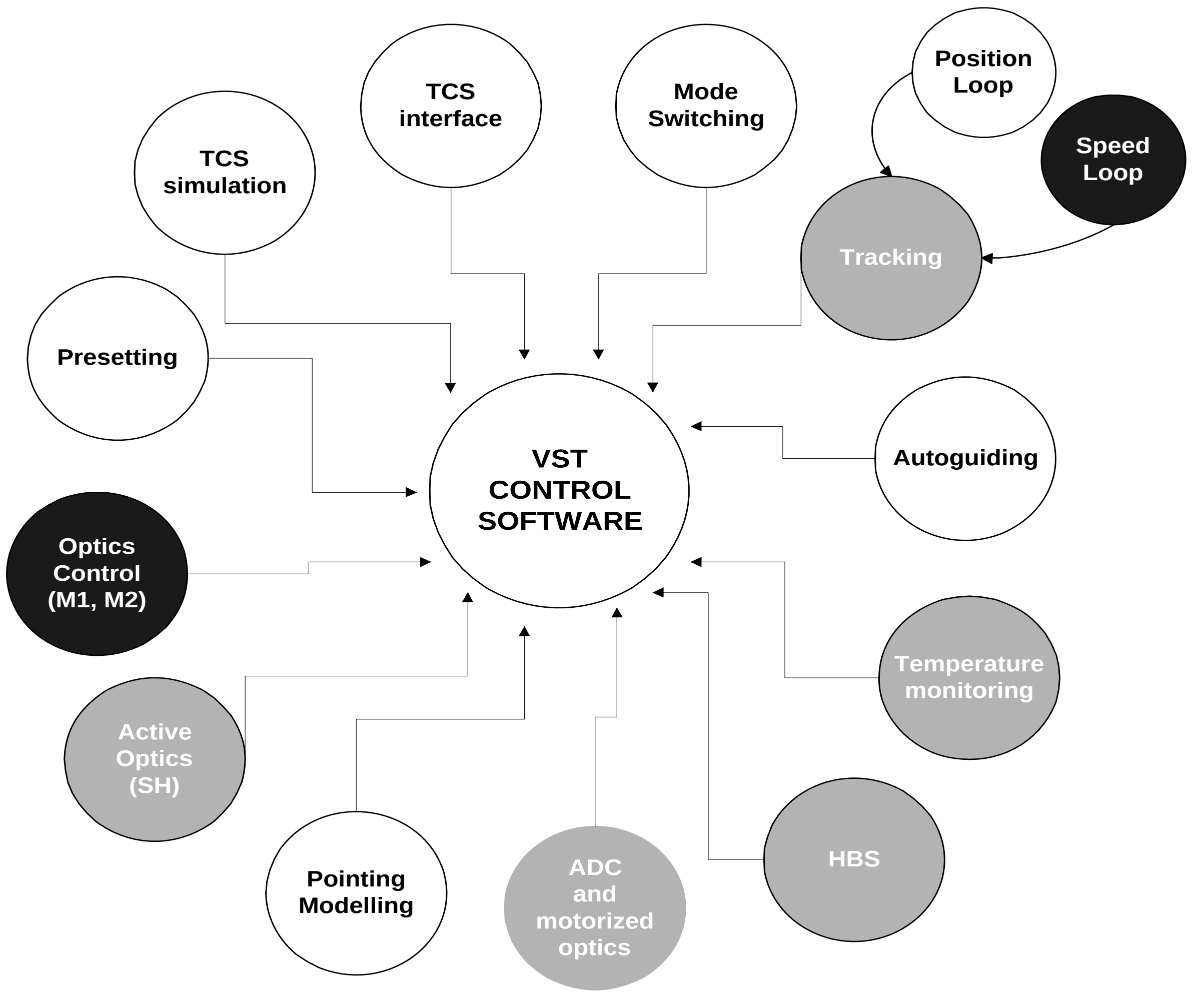
Fig. 5.2 - VST Telescope Control Software (TCS) Structure Overview

The functions of TCS are performed by a number of subsystems with LCU's and modules on higher level. The TCS makes extensive use of VLT Common Software, which provides common services, utilities and tools. In particular, generic services like logging, error handling and alarm handling are implemented by the VLT Common Software. The VLT TCS has already been applied to the NTT telescope. It is a general enough software package that can be reused on the VST to a large extent. Differences will exist at a lower level in the interfaces with some control electronics. Therefore there is a commitment by the VST team to re-use as much as possible, in agreement with ESO, existing VLT TCS control software. The functional scheme of TCS foreseen on VST telescope is illustrated in Fig. 5.3, where, for each functional module, the level of compatibility between ESO and VST TCS is indicated. These differences, in terms of hardware and software solutions, will influence the quantity of ESO software re-use for each functional subsystem. The exact amount of changes in the VST software is based on the following VST software team strategy, according to ESO-VST agreement:

1. Acquisition of VLT Common Software and ESO TCS by VST software team.
2. Analysis, for each VST functional subsystem, of the admissibility and compatibility between the ESO and VST software solution, based on their particular electro-mechanical and functional features.
3. In all the subsystems where it has been pointed out the complete compatibility between the two solutions, the ESO software will be de facto re-used.
4. Whenever this compatibility is partial, a particular effort will be achieved in order to perform as much ESO software re-use as possible. The software modules where changes are needed, will anyway follow the ESO constrains, in terms of functionality and development.
5. Whenever there is a complete absence of compatibility, due to the evident functional and hardware diversity, the VST team solution will follow the main ESO constrains, but it will be based on the particular electromechanical requirements and strategies, already discussed and basically accepted during ESO and VST team technical meetings.

It is therefore important to underline that the following TCS basic design requirements will be taken into account by the VST Team:

- All the interfaces with the telescope will be based on the ESO standards.
- All the active controls on the telescope will be handled at the LCU level, following ESO standards.
- All the possible commands available to control the telescope actions will be based on ESO standards, in terms of their name, meaning, syntax and behaviour, according to the corresponding commands already implemented in the ESO TCS.
- The organization of on-line database, logging system, error reporting, message system, scan system, will be based to ESO standards, according the same implementation of ESO TCS.
- The direct communication between LCUs will be denied and, when needed, it will be always achieved through TCS WS. The only exception to this rule will be related to correction information coming from the guiding system, that must be delivered to the other axis control LCUs in real time, in order to respect the control loops timing.



○ full compatibility between ESO and OAC software

◐ possible or partial compatibility between ESO and OAC software

● NO compatibility between ESO and OAC software or embedded controlled

Fig. 5.3 - VST vs ESO Control Software Modules Layout

6.3.3.1.1 Definition of TCS users

The typical users of the VST TCS are:
Observers
- Operation staff

Development and maintenance staff
Other control software packages

6.3.3.2 TCS Software Levels

The higher level TCS software is mainly implemented on a coordinating level, above the LCUs, but in some cases there might also be higher level software in a LCU. In other words, the term "software level" refers not to where the software is running, but the functional, coordinating nature of it, (Fig. 5.2). The modules at the lowest level of the layout will be provided by VST software team following the requirements and standards suggested by ESO and re-using ESO software, but in some cases they will require some modifications with respect to VLT Common Software standards, (Fig. 5.3). From the functional point of view, the TCS will consist mainly of the following modules:

Pointing modelling
Presetting
Tracking
Autoguiding
Active optics
Optics
Mode switching
HBS monitoring
Temperature monitoring
Engineering and maintenance

For each of them, a preliminary detailed design will be provided in the following.

6.3.3.3 TCS Included Subsystems

A typical VST TCS subsystem is made up of:

The opto-mechanical equipment to be controlled.
One LCU with LCC software.
Subsystem specific application software in the LCU.
Possible subsystem specific application software outside the LCU, but handled by the LCU.

Below is a list of subsystems included in the scope of VST TCS:

Altitude: position control of the telescope altitude axis.
Azimuth: position control of the telescope azimuth axis.
Rotator: position control of the telescope rotator axis.
M1: control of axial support of primary mirror.
M2: control of secondary mirror centering and focusing.
Auxiliary motorized optics, (ADC, shutter and filter wheel, group of lenses correctors).
Shack Hartmann analysis system.

6.3.3.4 Main Axes Control Software Design

6.3.3.4.1 Overview

This section contains a preliminary description of functional specification about the main axes, azimuth and altitude, control software, required for the VST Control Software design phase. The software will run on dedicated LCUs and execute commands from TCS higher level modules. This description should also be considered quite flexible, because at level of preliminary design. Therefore the software functional description may change to reflect hardware strategies evolution. It is foreseen the use of one LCU for each axis. Each axis is provided with own control cabinet equipped with the main boards, respectively, for digital/analog I/O lines control, time reference synchronization, motor control, power amplifiers and cooling control. The software running on LCUs is a low level part of TCS interfaced only with the local axis to control. Axes are considered as independent. The correlated motions of the axis, when the telescope is tracking an object, are achieved through a very accurate time synchronization of the LCUs. As issued in the VLT Common Software documentation, standard CCS-LCC mechanisms, such as msg system, database, scan system and alarms are used to interface the Alt/Az software with TCS. The software devoted to control the VST altitude and azimuth axes performs the following tasks:

Control and monitoring of the altitude axis for pointing and tracking.
Control and monitoring of the azimuth axis for pointing and tracking.
Providing of facilities to tune performances of the hardware controlling the axis.
Providing of test facilities for maintenance operations on the hardware controlling the axis.

The basic purpose of the VST axis control module is to drive the tracking axis under position control to follow the apparent motion of an astronomical object in the axis coordinates. The coordinate transformation calculations are executed by a particular sub-module which passes the result in the axis coordinate system regularly to the axis control module. These values are taken as reference points for an internal interpolation of the position trajectory in time where the axis has to be positioned. In order to perform this task, the axis control module has to perform the following main operations:

Read the encoder system.
Read the UT time.
Switch on/off the drive.
Close a position control loop.
Provide the standard commands to control the system.
Monitoring error conditions.
Handling of error conditions if they occur.
Monitoring interlocks.
Handling of interlock signals if they occur.
Control of the encoder system including start-up and calibration.
Control of the drive subsystem and interfacing to all HW I/O.
Provide commands for tuning, test and maintenance.

The VST axis control module is completely residing on the LCU as it has to meet real-time requirements. It will consist of the following software modules:

A module containing all the hardware and software common definitions used by other modules.
A module containing the control logic to position an axis with a dedicated position controller. It must also provide the standard commands necessary to start, stop and move an axis in pointing and tracking mode. Additionally it will include a set of maintenance functions which allows to preset and move an axis in speed mode (i.e. without position control loop).
A module providing the access to the axis encoder subsystem, needed to read, initialize and calibrate the encoder.
A module providing the access to the drive subsystem, the interlocks, proximity limit switches, monitoring signals and other hardware components of the axis. It mainly provides functions for:
Switch on the drive.
Switch off the drive.
Report the status of the interlocks.
Report the status of the vicinity limit switches.
Perform cyclic operations.

6.3.3.4.2 System Functional Design

The main features of the VST axis control module design is to provide a generic module containing:

The control.
The position controller.
The standard commands for driving a position controlled axis.

These functionalities are completely independent on the encoders used and on the particular signals and on the hardware used to drive motors, control interlocks or other I/O signals. A preliminary list of the VST axis control module features is here reported:

PI controller for pointing and tracking errors.
Support of emergency limit interlocks and proximity switches.
Support of speed mode.
Support of status database attributes.
Maintenance commands to preset and move in speed mode.
Maintenance commands to start-up and calibrate an encoder system.

The main function of the axis control module is to control the corresponding axis in order to allow astronomical observations, namely pointing and tracking. Auxiliary functions and facilities are provided to support performance tuning and maintenance operations. Normal operations involve monitoring and control of the following hardware elements:

Encoder
Amplifier
Velocity controller
Interlocks
LCU thermal control
Motors
Brakes

The control hardware section of the whole VST PDR document provides a functional description of the hardware interfaces with these elements. During normal operations, the position control of the axis is performed by a standard PI (Proportional Integral) servo loop. The servo loop is fed with reference positions calculated from the astronomical coordinates of the target at different instant of time. Before being passed to the servo loop, the information about position references is checked in order to avoid hardware limitations in terms of speed, acceleration and jerk. The shaped reference position is checked against the current position read from encoders and a velocity reference value is calculated from the measured differences. This velocity is applied to the hardware tachometer velocity controller in order to produce a torque signal to control the power amplifiers driving the motors of the axis. The states, described in the Mode Switching section of this document, applied to VST Altitude/Azimuth control LCUs can be summarized as follows:

OFF: axis LCU not operational. No telemetry can be returned from LCU.
LOADED: axis control software is initialized and ready to initialize the hardware.
STAND-BY: axis control software is completely booted and the database totally configured. In this state the amplifiers are off, no power connected to motors, brakes are engaged. Both velocity and position control loops are opened.
ON-LINE: in this state the amplifiers are switched on, power is connected to motors, brakes are released and velocity control loop is closed. Relatively to position control loop, the following sub-states can occur:

PAUSED: in this sub-state position control loop is opened. During this state, the axis can be moved only under velocity control loop. During these operations, the software must take care of the hardware limitations in terms of speed, acceleration and jerk. This sub-state is normally achieved to initialize the encoder or for maintenance operations on the axis.
IDLE: in this state position control loop is closed, axis is completely under control.
POINTING: the axis is moving towards a new target position not yet reached.
TRACKING: the axis has reached the target and is following it. The axis remains in this state until a new pointing command is sent from TCS. When operations are finished, the system is brought back to STAND-BY state with standard LCC command.

STAND-ALONE: this state is the same as ON-LINE but it is dedicated to maintenance and test operations, so the axis is not under control of TCS.

Normal transitions between the above states is under responsibility of TCS, except for the STAND-ALONE. Other special state transitions, (e.g. safe transitions in case of emergency), are achieved by automatic independent procedures. Between these states some basic functions operate on the individual hardware elements. Here there is a preliminary list of basic control functions for VST axis:

The hardware devices of the VST are self protected through a software independent chain of interlocks. The LCU software monitors them to identify activated interlocks. Enabling the interlock chain is the normal way to startup the telescope. If everything is ok, the hardware automatically switches on the power amplifiers and disengages the brakes. In case of failure, brakes are automatically re-engaged and amplifiers switched off. Of course, VST control software must monitor the interlocks to identify possible failures and return in case of trouble to a software configuration allowing a safe restart of operations.

The velocity controller includes velocity commands, normally directly issued by the position servo loop. The tachometer status is monitored and reflected in the on-line database.

The torque reference signal delivered by the velocity controller to the amplifiers is monitored and reflected in the on-line database, as well as the actual torque signal of each amplifier. When the encoder is initialized, the amplifier is instructed to use the encoder data available. Amplifiers are automatically switched on when the interlock chain is enabled, and automatically switched off in case of troubles. Their status is monitored by the LCU.

Encoder reading is controlled and performed from within the position servo loop. The status of the encoder and the position value are made available to other applications via the on-line database. However, considering the reading rate, only a subset of the values can be made accessible. Except during calibration procedure, encoder reading must be triggered regularly as the encoder value is also required by the motors.
Temperature sensors of motors are monitored and stored in the on-line database. They will be used to generate alarm notifications on the WS.

The interlocks are monitored and stored in the on-line database. They will be used to generate alarm notifications on the WS. Some of them are: operational limit out of range, hydraulic oil bearing fault etc. This list will be updated when the hardware design will be completed.

There are also combined functions that provide the functionality required for astronomical observation. They are the start up and shut down of the axis; pointing the axis to a given target position (pointing); following a moving target position (tracking); monitoring of the hardware and reacting on abnormal conditions; maintenance of the database in terms of a physical and logical image of the controlled hardware.

It follows a preliminary list of functions involved during VST main axis control operations:

Initialization functions

Start up: this function brings the telescope axis from STAND-BY to one of STAND-ALONE, ON-LINE and IDLE states. A particular attention must be taken on encoder status before switching the state. If the encoder is not initialized, the procedure first goes to PAUSE state, performs the encoder initialization and then switches to IDLE state.

Shutdown: this operation brings the telescope from ON-LINE or STAND-ALONE states to the previous STAND-BY state. The encoder can be left initialized.

Pointing functions

Pointing is the action of giving a new target position to the axis. This can be done in two ways: absolute positioning, with a real pointing command from reference position, or relative positioning, with an offset command from actual axis position. In order to perform the pointing operations, two types of object targets can be handled, depending by the scope of the pointing action:

Fixed target: in this case the pointing operation is achieved with the simple scope of testing positioning of the axis. The coordinates of the target are given in Alt/Az units. The axis will move to the target position, calculated from the given offset and the actual position. Once it has reached the target position the axis control module switches into the IDLE state.

Moving target: in this case the pointing operation is performed with the scope of tracking the object. The coordinates are given in alpha/delta format. Based on absolute time reference and astrometry calculations, these values are then converted to Alt/Az coordinates, cyclically recalculated and used for the target positioning of the axis. Once the axis is close to the moving target, the axis control module switches into the TRACKING state.

Of course, the pointing operations can be achieved from STAND-ALONE or ON-LINE states. Finally, as discussed, from POINTING state the module can switch into the IDLE or TRACKING states. A particular procedure will be provided for VST azimuth axis during pointing actions: the telescope motion to reach the target will be optimized according to the duration parameter, (short pointing time, short pointing time compatible with a minimum tracking time, long tracking time). Anyway, the requested position is checked against possible limits. In case of invalid pointing command, no change is performed in terms of operational states and a severe error is returned. Again, if the pointing command contains an invalid tracking time command, a warning is issued during the pointing operation, reporting the remaining tracking time available.

Tracking functions

After a pointing command the axis will only accept another pointing command or command related to the actual target.

Abort or Stop: after this command the axis will stop tracking and will go to IDLE mode at the actual Alt/Az position.
Modify Velocity: this function is used to change on-line the actual axis speed during tracking, in order to correct the position of the telescope.
Read remaining tracking time: this function is used to know the information about the amount of time remained for actual observing operation.
Offset: with this command a position offset is given to tracking coordinates in order to change the reference position.

Maintenance functions
In the VST axis control module it is foreseen an ulterior set of functions, used during maintenance operations:

Calibrate encoders: the calibration software for the encoders is embedded into the encoder units. When a calibration procedure is achieved for encoders, the axis control module is switched to PAUSE mode. During this phase the axis can be moved under velocity control loop.
Pointing: for the VST maintenance phase there are some special axis positions, reflected in the database, that can be reached with the presetting and pointing commands, as discussed above.
Tuning: some tuning operations on velocity and position loops can be achieved at the maintenance level.
Read position: this function reads the actual axis position given by the encoder. This function returns the last position value of the position loop, without reading directly from the encoder, to avoid disturbances.
Read velocity: it returns the velocity command actually applied to the velocity controller.
Move axis: this function is used to move the axis to arbitrary positions within the limits.
Read signals: this set of commands returns the value of many types of signals, used mainly to control the axis status. As said before, these signals are normally monitored at run-time during normal operations, but, if this monitoring is not available, the reading can be achieved polling directly from the hardware. They are :

Interlock signals
Brakes signals
Motor signals
Tachometer signals
Power amplifier signals

6.3.3.4.3 Error Cases

The general philosophy behind the error handling for the VST axis control module is that any parameter to commands, which is not within the correct range, lead to a rejection of the command. A severe error message will be generated and sent back to the TCS, for example if wrong coordinates are given which provide illegal positions of the axis. Each axis in the VST, infact, has a defined range of valid operating positions, often splitted into two sets, upper and lower limits. For the VST three levels of limits are foreseen:

Software limits: when these limits are crossed, the axis control mode is switched to IDLE mode, waiting for commands. In this situation, only position commands which move in direction away from the limit are allowed.
Proximity switch limits: when this limit is crossed, the axis is stopped and switched to IDLE mode. In this situation only commands which move in direction away from the limit are allowed.
Hardware interlocks: the axis is automatically stopped by hardware whenever this switch is reached. In this case the main immediate actions to be done are halting motors, engage of brakes and disabling power amplifiers.

From this position a movement back to operational positions is only possible by overriding hardware safety circuits. The Alt/Az control software switches then to STAND-BY mode.

Some malfunctions of the hardware can generate interlock signals also. For instance:

Motor current over temperature
Velocity controller interlocks
HBS warning/fault
M1 interlock
M2 interlock
General warning/fault

Interlock errors result in instant halting of the axis and switching to STAND-BY mode. The present list will be updated when the VST hardware layout will be frozen. The VST can still be operated in case of degraded mode switching, that essentially means the following cases: encoder calibration not performed or no active pointing modeling. These degraded mode of operations will prevent from reaching the full level of tracking accuracy. In this case an alarm will be issued. There are also limits on the temperature offset that can be used. If they are exceeded the command returns an error. If the cooling task notices a malfunction in the cooling process, an alarm will be issued. Possible causes can be:

Too low level of cooling liquid flow
Cooling liquid valve not working
Malfunction of the sensors

In this event case the fans of LCU will be able to guarantee the avoiding of damage of the hardware, but an alarm must be issued. This alarm must be used to trigger a transition of the axis to the IDLE safe state before the hardware switches off.

6.3.3.4.4 System Interfaces

During normal tracking operations, no direct interface between axis control module and users takes place. The user interacts with TCS which is responsible of the communication with LCUs. It is therefore important at this point to clarify that, following particular ESO sw constrain, no direct communication interface between LCUs is achieved. The only exception to this requirement is related to the situation where, during tracking phase, the guiding control LCU must deliver axis corrections directly to axis control LCUs in order to guarantee the correct axis control loop timing without potential delay. Special engineering user interface will be used during maintenance and test operations, (see Engineering and Maintenance section of this document). The axis control module has also to interface with TCS via the standard command interface with alt/az LCUs. With other parts of the system there are several levels of interface, for example, with the hydraulic system the interface is restricted to hydraulic oil bearing system interlock; with the ADC system the interface is based on the distribution of astronomical computations and axis position to ADC controller from axis control LCUs.

6.3.3.4.5 *System Data*

For the VST axis control module system data, all configuration and status information will be kept in the LCU database and transmitted to the TCS WS via scan system, where they are used for telemetry and diagnostic purposes. The LCU database structure and configuration will be realized strictly according to the ESO VLT Common Software recommendations. Its main structure related to alt/az device data branches will be the following:

Encoder
Power amplifiers
Velocity controller
Motors
Brakes
Interlocks
Logical representation of the axis itself
TCS specific data for coordinate transformations

Other information on system data will be added in the future, according to VST hardware and electronics final design choices.

6.3.3.4.6 *Software Control Loops*

During telescope normal operations, the axes control is obtained with digital control loops. The reference positions to the servo loop are the astronomical coordinates of the target, calculated at different instants of time. The reference position values are compared to the actual position values read by encoders, providing an error signal which is used as an input to the speed controller. It is compared to the actual velocity read by the tachometer and the measured difference, i.e. the velocity error, is sent to the speed controller, which provides the proper control action to the actuators through D/A converters. For the three axes position loops, the control software will be running on LCU related to appropriate axis, and it will be based on position interpolation through encoder reading feedback. Many efforts have been made in the software design optimization in order to avoid intrinsic encoder error effects:

Definition of dynamical LUT for each encoder unit for gear coupling error compensation and for medium frequency errors. The LUT will be dynamically defined on the basis of on-line analysis of the instantaneous encoder position readouts with respect to the readouts averaged values of the remaining units.
Moving average of the last position readouts to reduce the interpolation noise and the high frequency encoder errors.

The velocity control loop for rotator axis will be based on MACCON motor control device, interfacing directly with tachometer, and therefore handled directly via hardware. For the altitude and azimuth velocity control loop, a dedicated control unit with front-end electronics will be provided. It will consists of a commercial standard PCI-bus CPU, connected with standard serial or parallel line to related LCU. The customized software, needed for this controller will be included in the TCS software module that, basically, will be based on the ESO TCS module implementation. The software for preloading, velocity control loop, cooling and power control, will be resident on a DISK-ON-CHIP device installed on the dedicated control unit and it will automatically come up when the axis control system will be loaded from WS. The main axes incorporate two brushless motors controlled in pairs and a torque preload is applied between motor couples in order to compensate for backlash in the telescope gearboxes. The preload torque function is obtained by means of an adaptive software algorithm embedded in the control system that regulates each motor preload torque value according to the instantaneous axis acceleration and/or instantaneous total torque requested. In the velocity control loop the tachometer acquisition system will use a standard 16 bit A/D converter. The special embedded controller is used to automate the tachometer data acquisition. It determines the appropriate number of samples to be provided. An oversampling technique is the solution adopted to reject tachometer noise. The dedicated CPU will collect the data, provide tacho oversampling, generate an average of the two tacho readouts, and provide data to the speed loop every 2 ms. This composite average represents the velocity feedback used by the velocity servo loop. The synchronization between position and speed control loops will be based on the availability of encoder position reference during 500 Hz operation position loop. The main advantage of the usage of an embedded controller is the avoiding the overloading of LCU and the major flexibility in terms of control parameter tuning. The main features of this system will be:

Stand-alone device, where it is possible to test the axis control with a direct dedicated interface;
No external storing devices, (floppies, hard disks), are required to store and load the software. All control software will be resident on special DISK-ON-CHIP, ready at startup of the system;
It can be initialized and handled directly by LCU;
It will consists of standard RS232/CENTRONICS communication bus for diagnostic and telemetry;
The velocity control loop is completely digital, giving much more flexibility and long time reliability.
Synchronized preload and velocity control loop provided by dedicated local control software;
Collection of diagnostic and telemetry data related to the axis

In order to achieve the level of performances required for axis control loops, the implementation of the VST axis control software will :

Take care that the quantization effect due to the position servo loop is low enough. This effect can be reasonably rejected if the position servo loop operates at a high enough frequency. A typical formula is to use a value 100 times the system bandwidth. For the VST we have also the lower bound due to the encoder reading time, (1 ms), so a minimum of 2 ms is achievable.
Avoid undersampling in the trajectory generator.

To assure accuracy of reference values passed to the servo loop, the following checks will be performed:

The encoder calibration procedure has been successfully done.
An accurate pointing and tracking tuning has been established and installed on the related axis LCUs.

In these considerations, it is assumed that all other effects not under control of the axis control software will be taken into account by the relevant control hardware. A preliminary list of the effects considered is the following:

Static friction
Encoder accuracy and quantization
Tacho ripple and hysteresis
Motor torque ripple and cogging

6.3.3.4.7 Presetting

6.3.3.4.7.1 Introduction

This section takes care of functions requested to move VST telescope to a new object or a new fixed position. The presetting operation requires that the absolute coordinates of the object or of the fixed position have to be passed to TCS. The presetting subsystem foreseen for the VST will be based on the presetting module already implemented in the ESO TCS software.

6.3.3.4.7.2 Telescope Pointing modeling

This section includes the functions that creates and updates pointing models, automatically or interactively. The term Pointing Model is referred to both pointing characteristics and related coefficient values. The Pointing Modeling strategy followed in the VST, will be based on ESO VLT pointing modeling philosophy and implementation. It is therefore obvious that some little changes are needed. The main reason is that VLT telescope has more focus stations, more pointing correction terms and a completely different guiding system. Furthermore, the intrinsic feature of the VST, (i.e. survey telescope, one focus station), makes relatively easier the pointing model design. During normal operations there is always a pointing model active, named "current telescope model", embedded in the TCS software module, which will be used when the telescope is set up for observation in the Cassegrain focus station. The total functionality of Pointing Modeling can be divided in 4 groups of functions, each one subdivided in sub-functions:

- **Measurement**
  1. direct
  2. centering-object-first
- **Calculation**
  1. small update, fixed model configuration
  2. total update, fixed model configuration
  3. interactive analysis
  4. optional: total automatic update, optimizing model
- **Modification**
  1. modify configuration file
  2. store configuration file
- **On-line implementation**
  1. full library available

For all the above functions, except for the On-line implementation, there are commands available that allow an interactive, step-wise execution of the pointing modeling. There are also commands which execute a full automatic sequence. For the **Measurement** group of functions, the telescope is preset to a star selected from pointing catalogue. Pointing measurements can be done by use of the guiding CCDs, by calculation of the center of rotation of the rotator axis and taking this position as the reference point for pointing measurements. The sub-pixel accuracy of the star centroid position is actually TBD. The output of the measurement functions is a file containing nominal and actual star positions, from which position errors can later be calculated, and all information about the model used during measurements. There are, as described above, two different procedures for the measurement:

1. **direct:** the star position in the field, as it happens to be after a telescope preset, is measured. This is the normal way, used by the automatic functions.

2. **Centering-object-first:** the telescope position is corrected, so as to put the star on the reference point. This might need a manual correction, if the pointing is so bad that the star does not even appear in the guiding CCDs field.

For the **Calculation** group of functions, it is possible to define the following main group:

1. **Base line functions:** as discussed above, there are two different calculation schemes, where a predefined pointing model configuration is used: small and total model update. Each of these schemes is included in one of the automatic cases. Additionally, there is an interactive update mode, in which the user defines which terms will be used. The output of the Calculation function is a file containing names and change of values for the pointing terms in the used configuration, and all information about the model used during measurements.

The **Modification** functions create a new, complete model file, which is a previous model file updated with a set of calculated changes. In case of a total measurement, the complete model is updated.

The **On-line implementation** takes care of pointing corrections during on-line tracking. All the possible pointing correction terms that can be used to design a new model are also implemented for on-line compensation. Infact, it is reasonable to correct the reference positions, calculated for tracking, using the pointing model. In this case the correction is done every cycle, i.e. for each calculated reference value. The coefficients of the correction terms cannot be modified individually. They can only be re-calculated and re-installed by explicit Pointing Modeling commands. These commands modify the position of the pointing axis, which is defined as "the point in the instrument focal plane that corresponds to the actual telescope coordinates". The function uses the "current telescope model", defined at start-up, to decide which terms are actually used. The On-line implementation is anyway static, in the sense that no coding/compilation or linking has to be done when a new model, (i.e. new values and new terms), has been designed.

Following the above Pointing Modeling philosophy, a number of commands are available for small and total model change/measurement. The main difference between the two is, of course, a matter of the time it takes to do the measurements, i.e. how many stars are necessary for the modeling. The small model needs only few stars, so it can be done at almost any time. The total model needs much more time to perform the measurements, (for example, it is reasonable to think that the total model requires a whole observing night for its measurements), and the stars used must be automatically selectable by dedicated functions, taking into account an optimum distribution in the sky. A preliminary list of suitable commands is defined below:

1. **Set small/total model:** this function automatically makes presets and measures the pointing errors for a number of stars, according to pre-defined parameters. When the sequence is ready, a complete, new model file, updated for a small/total change, is calculated and stored. The model terms to be updated is pre-defined for each case.
2. **Make small/total model:** this function automatically makes presets and measures the pointing errors for a number of stars, according to pre-defined parameters. But in this case, the output is a measurements file. No model update is made.
3. **Make model change from given measurements:** this function uses requested measurement file as input for calculation and installation of small/total model change.
4. **Initialize pointing measurements:** this function makes the necessary preparations for measurements of direct or centering-object-first style, as requested.
5. **Switch on/off automatic storage of measurements:** this function is used to select if storage of measured star positions should be done manually or automatically.
6. **Store actual position in pointing measurement file:** this function is used when manual storing is switched on.
7. **Insert/delete model term in/from model configuration:** this function is used during interactive analysis and model calculation. The function does not change the active model.
8. **Calculate model change:** this function uses requested measurement file as input for calculation of model change for a temporary, interactive model. Result is stored in temporary file only.
9. **Define current pointing model:** this function is used to switch between models.

10. **Make correction for systematic errors:** this function can be used when the pointing operation has a more or less well known systematic error. The function adds the requested values to the pointing corrections.
11. **Measure the position of a pointing axis:** this function is used to measure the distance between the default pointing reference position and some other position in the focal plane of the instrument. The object is centered on the wanted position, using the guiding CCDs and then this function is called. The position is measured and passed back to the caller.

6.3.3.4.7.3 Basic Presetting Functions

All the functions foreseen in this module are based on command delivering, which means that there are no automatic procedures for presetting, nor that presetting can be achieved by some TCS internal functions. Anyway it is possible to enclose groups of commands into presetting script sequences that of course will appear as hidden to users. All presetting commands using sky object coordinates, when accepted, will activate tracking immediately. This means that, for observation reliability, the astrometric corrections of axes position, used during tracking, will start from the beginning of presetting execution. Therefore, all presetting commands, made using fixed axis position coordinates, will not activate the tracking operation. Usually, the delivering to TCS of presetting command will be preceded by the definition and assembling of user's targets coordinates and associated guide stars. This preprocessing action will be performed through a special loading command. For the VST presetting operations, the target object definition can assume the following types:

- Coordinates given in a setup file. RA and DEC must be required parameters.
- Coordinates given in catalogue.
- Absolute position, i.e. ALT and AZ angles.
- A predefined name used for special positions, e.g. ZENITH.

More specifically, the VST is requested to move to a new object or sky zone by a setup file command, with target information specified, or by a direct command. In normal conditions, its behaviour is as follow: right ascension and declination coordinates must be specified together with other optional information. It is TBD the definition of the final catalogue used for coordinates. The given coordinates are converted to actual telescope coordinates, the telescope is positioned according to that, the rotator is set to its default or requested orientation and the tracking is activated. Optionally, the motors for ADC correctors can be activated automatically in order to perform the atmospheric dispersion corrections.

6.3.3.4.7.4 Presetting Commands

Here a preliminary list of commands, applicable for operations on Cassegrain focus is described:

SETUP: this is a function that loads and processes the presetting setup files, where a set of parameters is included, such as RA and DEC, epoch, equinox, proper motion alpha, proper motion delta, radial velocity, wavelength, parallax, or coordinates given by name, (object catalogue name, object id), or absolute position in terms of direct AZ and ALT angles. Optional are the ADC settings, such as RA and DEC for optimum setting, zenith distance for optical setting.
Save current SETUP: this function saves the complete current TCS setup file.
Clear current SETUP: this function clears all parameters in the current TCS setup file.
Preset to given coordinates: this function is used to furnish coordinate parameters directly.
Define default object catalogue: it is used to specify the name of the catalogue to be used.
Preset telescope to given ALT and AZ position: this command can be used to set the VST to any predefined axis position. This function will not involve any movement of the rotator axis.
Set telescope to moving object: from given coordinates, telescope position and tracking velocities are calculated, the telescope is pointing and tracking start.
Stop telescope movement: this function is used to stop telescope motion with position closed loop.

6.3.3.4.7.5 Presetting Error Cases

For the VST presetting command module, any formal and/or format error in command or parameters results in the rejection of the command, with error reply. Particular case is the occurrence of illegal positions. In this case, if a requested new position would require an illegal telescope setting, obviously the presetting command will be rejected and an error reply will be reported. The following are illegal positions:

Altitude angle too low: the software horizon limit was reached or interlock limit alarm occurred.
Altitude angle too high: this limit is not checked for a fixed position.

6.3.3.4.8 Tracking

This section addresses the preliminary design of the tracking module foreseen for the VST TCS. The basic purpose of the tracking module is to drive the telescope main axes to follow the apparent motion of objects. These operations involve mainly the azimuth, altitude and rotator axes position control and possibly the ADC control.

6.3.3.4.8.1 Overview

The preliminary design of the tracking module was derived from ESO VLT Common Software concept. In the following figure (Fig. 5.4), there is a diagram layout showing the relations between tracking module with externals and the workflow between WS and LCU levels.

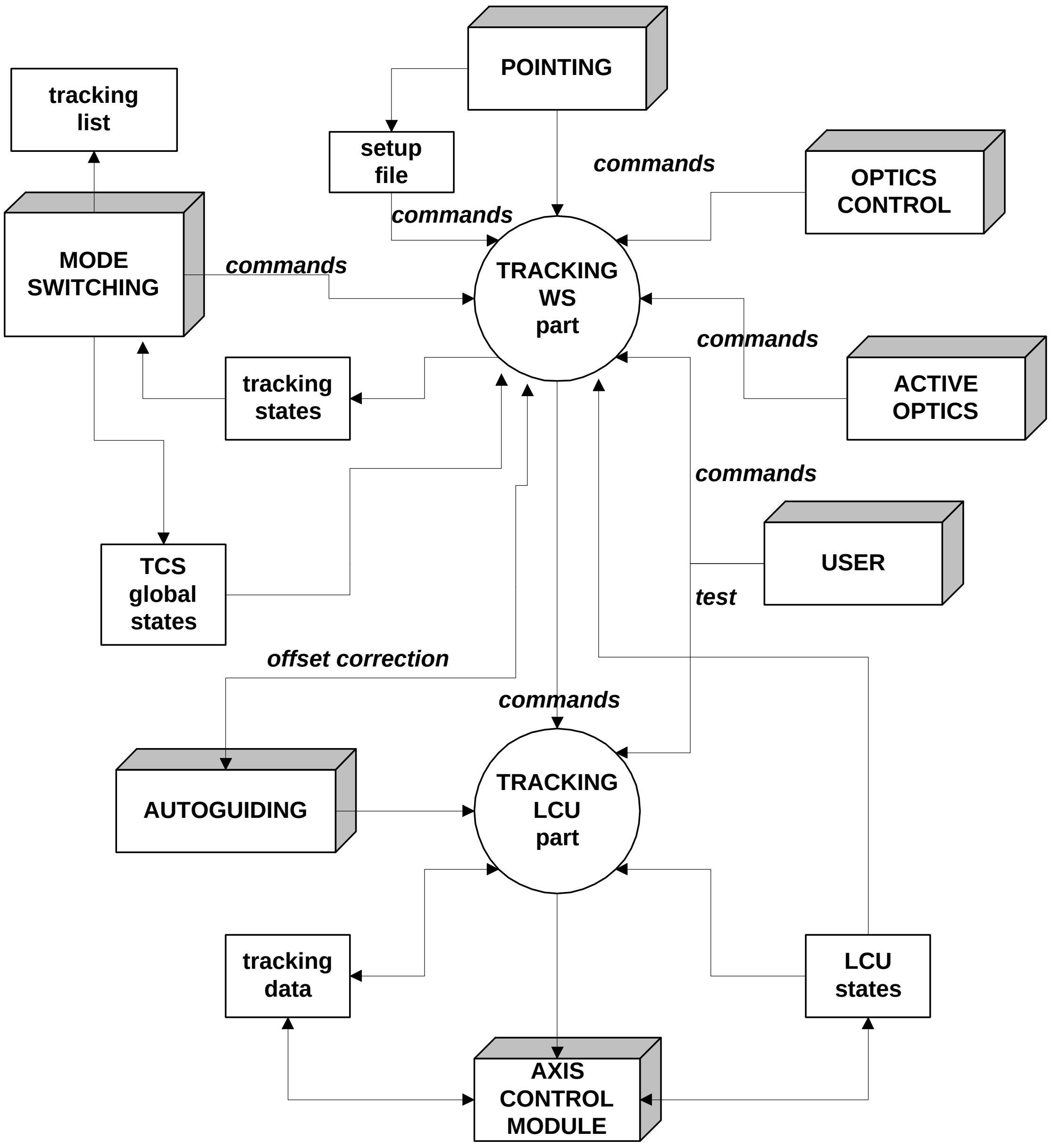


Fig. 5.4 - VST Tracking WS-LCU design diagram

During tracking operations there is a continuous data/commands flow between WS and LCUs. The run-time state of the tracking module is stored and reflected in the on-line database. A preliminary discussion about tracking states is the following:

OFF: in this state all LCU and tracking processes states are undefined. This operational mode can be the result of following sub-states:
STEADY: in this case the tracking module is in OFF state after a normal bootstrap or shutdown.
ERROR: in this case the OFF state was resulting after an abnormal situation during tracking process.

STAND-BY: during this mode all LCUs involved in the tracking functions are ready with software loaded. Also in this state there are two sub-states, STEADY and ERROR, that indicate, respectively, normal or abnormal transitions to STAND-BY state.

ONLINE: This is the case where the tracking module is operative. There are several sub-states depending on the particular situation of axes:

IDLE: the telescope and rotator are not moving, but position loops are closed and brakes are disengaged.
TRACKING: the telescope and rotator are tracking.
POINTING: this is a transition sub-state, used when the telescope is moving to a new object after a pointing command or an offset command.

The VST tracking module will consists of a tracking controller process, able to receive all external commands and handles the related actions to be done.

6.3.3.4.8.2 Tracking Module Commands

A preliminary list of tracking commands, not including the ADC, accepted by the tracking controller process is the following:

Alt/Az coordinates: pointing of new alt/az position coordinates.
Object coordinates: pointing of new sky position coordinates.
Named position: pointing of a prefixed position.
Offset RA, DEC: defining of an offset step for RA and DEC.
Parameters correction: request for manual parameters correction.
Add velocity to AZ: addition of velocity for axis AZ.
Add velocity to EL: addition of velocity for axis EL.
Rotator offset angle: set new rotator offset with respect to the current position.
Get tracking time: get remaining tracking time.
Get AZ pos: get actual tracking AZ position.
Get EL pos: get actual tracking EL position.
Get ROT pos: get actual tracking ROTator position.
Set deltax, deltay: set instrument aperture position.
Switch to stand-by: switch the tracking module to stand-by state.
Switch to on-line: switch the tracking module to on-line state.
Switch to off: switch the tracking module to off state.
Get state: returns the state of the process.
Get status: return status of the process.
Start simulation: switch the tracking module in simulation mode.
Stop simulation: switch the tracking module from simulation mode.
Stop action: stop the current action.
Terminate: terminate the process.

For each of these commands, a callback is provided, with the following general format:

Receive the incoming parameters.
Check that they are correct.
Error reply if not correct.
Update WS database.

6.3.3.4.9 Autoguiding

6.3.3.4.9.1 Introduction

The autoguiding system, for an AltAz telescope, is able to furnish to the main axes control loops the feedback information about position corrections for errors of low frequency, i.e. < 1 Hz. Such errors can be caused by errors in coordinate or refraction calculations, flexure or vibration in the structure, temperature variations etc. Corrections are done to the telescope position, i.e. to the altitude, azimuth and rotator/adapter axis. The autoguiding besides is not supposed to correct for variations in atmospheric seeing, except for slow variations.

6.3.3.4.9.2 Autoguiding System description

For the VST autoguiding system design, the VST team has two basic choices:

1. the approach until now followed was to measure positions errors by means of four CCDs mounted on the instrument of the telescope, the OmegaCAM, a wide field CCD mosaic camera 16k x 16k, designed for the VST telescope Cassegrain focus, [26]. More in detail, following the idea to realize an autoguiding system as a guiding camera mounted in the instrument as slit viewer, in the VST telescope four slit viewers will be provided, by means of four CCDs located on the four sides of the instrument camera CCD. These four devices will be used for telescope guiding, focus adjustment and tip-tilt correction. The instrument dewar will contain also head electronics for the guide CCDs and cabling to the external guiding electronics. These four devices are located on the telescope instrument but, following ESO requirement, they will be handled by the VST TCS control software module.
2. The opposite approach can be to adopt the strategy followed by ESO for the VLT autoguiding system. This solution includes the use of a probe camera system in rotating coordinates. This solution seems to be better for VST for the following reasons:

- ➢ it will be possible to re-use a considerable amount of ESO hw and sw. This guarantees also a strong compatibility with ESO systems.
- ➢ it is not necessary to modify the control system of OmegaCAM in order to allow the handling of the 4 CCDs, for guiding, by VST TCS.

The final choice provided for the VST autoguiding system is still TBC.

6.3.3.4.9.3 Control interface

If the CCD slit viewers solution will be adopted, the acquisition electronics for the four guiding CCDs, [26], and the electronics in the dewar to the guide CCDs controllers are not under OAC VST team responsibility. Anyway, our TCS software module will handle the autoguiding control system calculating the position errors, in order to correct altitude and azimuth axis positions during telescope operations through the astrometry loop. Apart from the acquisition system finally implemented, that, in any case will be necessarily customized, the VST and ESO software modules related to autoguiding basic calculations and corrections can be considered quite similar. The idea is, therefore, to re-use as much as possible the ESO autoguiding software.

6.3.3.4.9.4 Autoguiding Operations

We have to distinguish between normal operations before and during observations. In the first case, to save observation time and to simplify operations, guide stars will be searched, selected, checked and transferred to the TCS before the start of the observations. For the second case, during observations, the autoguiding functions are normally initiated when a telescope preset is done, as a default action. A default sequence of autoguiding actions is then automatically activated. This means that, normally, no manual commands are needed for autoguiding at observation time. The default sequence is, [4]:

- automatic probe setting to guide star
  - if pre-selected guide stars not available, make an automatic catalogue search
  - set probe to guide star candidate
- automatic lock on guide star
  - determine guide star: if OK continue, else take next candidate
- automatic start of autoguiding
  - check start conditions: if OK continue, else stop sequence
  - start cyclic autoguiding functions

If this automatic sequence cannot be used, e.g. if it fails at some point, or if a different sequence of operations is requested, there are possibilities to control the autoguiding by explicit commands, on different levels and on different points in the sequence.

6.3.3.4.9.5 Basic Autoguiding Functions

This section provides a set of basic functions of which the autoguiding batch functions are composed. Also a number of manual functions are described, that will be used mainly in special cases, e.g. for maintenance and test purposes. At this point all the particular functions related to perform the acquisition of the object with the four CCDs are not reported, basically because the details concerning this topic are still TBD. In the present document we want only to mention the functions that will be more fully described after PDR. The list of functions is the following:

- **Search fo guide star in catalogue:** there are three different guide star search modes: pre-selection, automatic search and manual search
- **Guide star acquisition:** this function implies the handling of acquisition system through the four CCDs or probe camera (TBD).
- **Check start conditions:** this function includes the information acquisition communicating that: all subsystems are OK; there is a suitable guide star detected, the telescope is tracking, the rotator and dome is tracking, the M2 is enabled.
- **Cyclic autoguiding functions:** this group of batch functions contains: check conditions for continuing, measure position error, calculate correction to telescope position, request correction to telescope position, differential refraction compensation.
- **Utility functions:** guide star selection, check guide star position, image display, status requests.

6.3.3.4.9.6 Cyclic Autoguiding Functions

Once the observation is started, the autoguiding system runs on a cyclical base. This includes also the guide camera CCDs, which cyclically deliver position errors and brightness of the guide star. The request for cyclic delivery is set up once only. The CCDs controllers send back data without explicit request for every exposure. The basic actions performed in each cycle are:

- **Check conditions for continuing:** the conditions are the same as for start of autoguiding. In case of any error a warning is issued and the guiding is stopped.
- **Calculate average magnitude and FWHM:** averages are calculated and stored in the on-line database for the actual guide star.
- **Measure position error:** this is performed cyclically by the CCDs.
- **Calculate correction to telescope position:** the position error is converted to corrections in alt/az coordinates. The calculated corrections are stored in the database.
- **Request correction to telescope position:** the calculated alt/az corrections are passed as offset steps to the tracking function. This is achieved by direct communication of these values to other alt and az LCUs.
- **Differential refraction compensation:** to correct for differential movements between object and guide star, due to differences in both wavelength and position, the reference position of the guide star is "moved" by

software. This movement must also occasionally be adjusted with a real movement of the CCDs, when the guide star is moving out of the center part of the CCDs field.

6.3.3.4.9.7 Autoguiding Control Commands

As said above the autoguiding is normally started automatically as a part of the telescope presetting function. When this feature is enabled, all actions necessary to activate autoguiding are done automatically. However it must be possible to control also various autoguiding functions by means of a set of explicit sub-function commands, listed below:

1. **Enable autoguiding**
2. **Disable autoguiding**
3. **Start autoguiding**
4. **Stop autoguiding**
5. **Select active guide camera CCD**
6. **Define guide star**
7. **Display requested part of detector field, once or cyclically**
8. **Save displayed image**
9. **Set CCD to given RA, DEC**
10. **Set CCD to given catalogue position**
11. **Offset step given Δ(X), Δ(Y). X and Y are in camera (CCD pixels) coordinates**
12. **Start/stop moving guide CCDs (TBC).**
13. **Automatic catalogue search for guide star**
14. **Interactive selection of guide star in catalogue**
15. **Switch on/off orientation indications**
16. **Switch on/off automatic "start autoguiding" at telescope preset**
17. **Set SETUP parameters used when searching guide star in catalogue** (min and max magnitude etc)
18. **Set guide parameters** (cycle time, integration time, etc)
19. **Get guide star data** (guide star position, position error, estimated magnitude, etc)

6.3.3.4.9.8 Autoguiding Error Cases

Several kinds of errors can occur for autoguiding system, for example no suitable guide star founded, telescope, rotator or dome not in tracking. In all cases an error message is issued and the autoguiding is stopped. But if an autoguiding error occurs while telescope is in tracking phase, the telescope operation will not be stopped.

6.3.3.4.9.9 ADC Autoguiding

The ADC autoguiding control sw will be integrated to the VST autoguiding control software as a separate subsystem.

6.3.3.4.10 Active Optics

6.3.3.4.10.1 Introduction

The role of this section is to furnish a detailed description of our software strategy adopted in order to measure the image quality in the focus station used for observation, and based on these measurements, to calculate and request corrections to the VST M1 support forces and to the VST M2 centering and focusing. With particular reference to M1 primary mirror, we distinguish between gravity force correction and wave front shape correction, respectively called *passive optics* and *active optics*. In the first case, an off-line pointing of actuator positions is done; in the second case a wave front sensor is used in order to give active movement commands to actuators, proportional to local wave front distortion calculations. A detailed description, from control software point of view, of dedicated devices and subsystems is also provided.

6.3.3.4.10.2 Active Optics Control System Description

The VST active optics control system philosophy is based on active position corrections on both M1 primary and M2 secondary mirrors. The M1 active optics control system is based on mirror support plates control through actuators, handled through a dedicated front-end electronics device, essentially composed by three units, (see Fig. 5.5), respectively, unit I for Input flow/buffer communication from actuators, unit O for Output flow/buffer communication to actuators, and unit C for Control communication with actuators, such as single actuator addressing command. The handling operations on actuators is accomplished by a multiplexing communication system between front-end electronics device and actuators.

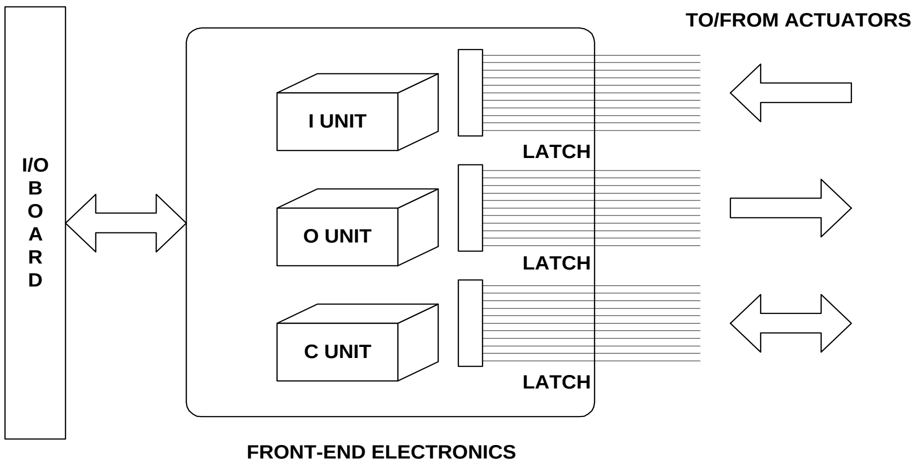


Fig. 5.5 - Actuator Control System layout for M1 active optics control

The M2 active optics control system is based on a dedicated electro-mechanical device, called Hexapod, that consists of a movable platform based on six linear actuators and driving electronics (see Fig. 5.6).

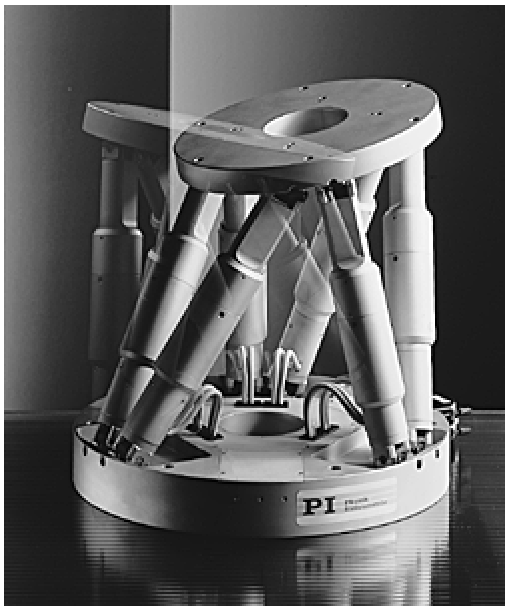


Fig. 5.6 - Hexapod device for M2 active optics control

All movements in six degrees of freedom are accomplished by DC-Motor driven linear actuators, representing the legs of the Hexapod system. The primary role of such a system is to correct all secondary mirror de-centering, de-focusing and coma-tilting phenomena during telescope operations. In order to perform these operations, the control system is able to transform all position commands given in Cartesian coordinates to Hexapod actuator axis specific positions and velocities characterized by relatively good accuracy. The Hexapod is controlled by a PC-based 6-axes DC-Motor Controller, (we call it Hexapod Controller), provided with an embedded controller. More specifically, the Hexapod Controller is based on a multi-processor architecture including a fast DSP motion control chip set, providing trajectory generation and closed loop digital servo control based on position information supplied by incremental encoders, and a host processor for communication and command handling. The operating software receives motion commands via serial RS-232 9.6 Kbaud communication from a host LCU. The control software is based on a driver, designed and developed by VST TWG software group, through SCPI language technology supplied by Physik Instrumente embedded into a high level C tool, including serial communication protocol. The VST M2 control software will be implemented from scratch, due to the unavailability of ESO VLT M2 software. However, the interfaces VST M2 will be compliant to the ESO VLT M2 standard.

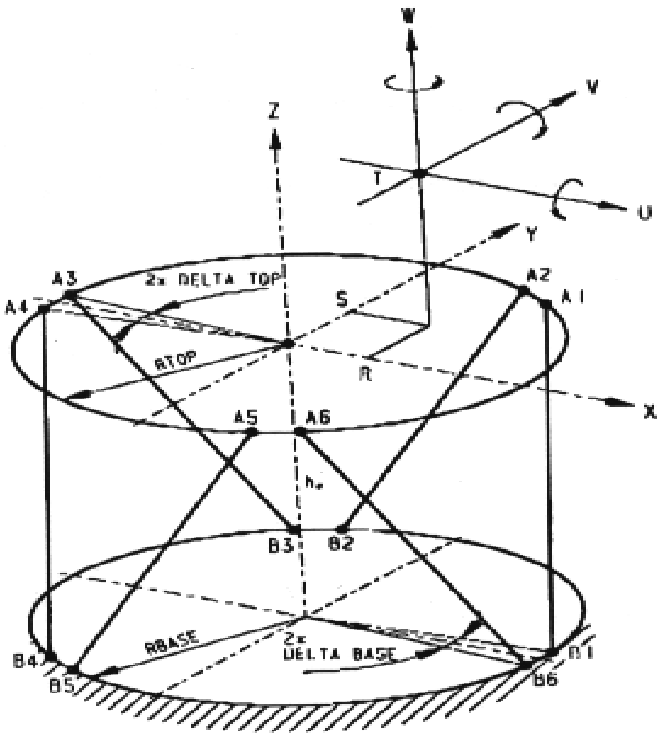


Fig. 5.7 - Hexapod coordinate system

The control software commands and coordinate transformations are based on the following definitions [25], (see Fig. 5.7) :

The Hexapod legs are mounted on the base plate at six points B1…B6. The set of six legs is arranged symmetrically on the base plate on a circle with the radius of RBASE;

The angle between the three joint pairs is 3x120° and the angle deviation from 120° is DELTABASE;

The opposed three joint-pairs A1-A6 are connected with the platform. They are located on a circle with the radius of RTOP and have an angular deviations from 120° named DELTATOP;

The origin of the fixed coordinate system XYZ is located at the center of the upper six joints A1…A6 and corresponds to the center of the Hexapod after initialization. All translations are performed relative to this point on an interpolated line. That means all legs start and stop their movements at the same time;

For translations, the orientation of the coordinate system is unchanged. For rotations a new pivot point can be defined with the coordinates R, S, T (not identical with the origin). Rotary axes are related to the new coordinate system u, v, w shifted from the origin by R, S, T. The coordinate system U, V, W is parallel shifted by translations in XYZ and also changes its orientation while tilting. After tilting u, the coordinate system UVW (not XYZ) also tilts and has a new orientation for tilting the next axis. In any case tilting will be performed in the sequence U, V, W. At the control software level the driver command structure complies with SCPI in its short form. All commands are transferred via the RS-232 interface. Communication parameters are 9600 baud, no parity, 8 bit, one stop. A standard zero modem cable will be used to connect the host with the controller. The serial protocol will be based on the append of special codes to SCPI commands, such as terminator line feed (#10#Ah), and related report messages. In terms of system accuracy, the Hexapod has resolution specifications that can be considered more than satisfying from secondary mirror control accuracy point of view. Anyway, the control software dedicated to Hexapod will provide, during calibration phase, a series of look-up tables, used to correct the intrinsic Hexapod structure flexures.

6.3.3.4.10.3 The Shack-Hartmann image analysis system

In order to furnish a feedback analysis tool for image quality control during active optics operations, for the VST it is foreseen the use of the SH wave front sensor, running on the LCU n. 3. The system adopted by ESO is not completely reproducible on the VST, because it is a survey telescope and, due to its optical configuration, the field dimensions don't permit to have sufficient room for the installation of a beam-splitter on-axis. The operative conditions foreseen during SH analysis are:

1. OFF-AXIS control and analysis in ON-LINE acquisition mode
2. ON-AXIS control and analysis in OFF-LINE acquisition mode

In the case 1), from software point of view, it will be necessary to generate accurate calibration tables with offset, in order to reproduce the ON-AXIS condition. The possibility will be therefore to implement the ESO software taking into account some changes and modifications essentially needed for the analysis subsystem. For the acquisition subsystem, infact, we think it will be possible to re-use the same hardware and software solutions adopted by ESO, (CCD, controller). From the control software point of view, the SH analysis requires the following input parameters:

- Maximum number of pixels allowed per spot
- Minimum number of pixels allowed per spot
- Side of area containing a spot (in pixels)
- Minimum number of brightest pixels to be used for centroiding
- Maximum number of brightest pixels to be used for centroiding
- Pixel size of the CCD chip (in microns)
- Threshold factor for detecting of the spots in the reference frame
- Eccentricity limit
- Threshold factor for mirror frame
- Eccentricity factor for mirror frame
- Diameter of primary mirror M1
- Diameter of secondary mirror M2
- Size of the central obscuration
- Focal length f1 of M1
- Focal length of the telescope
- Back focal distance beta in units of f1
- Conic coefficient k1 of M1
- Conic coefficient k2 of M2
- Distance M1-M2

- Distance M2-focal plane
- Maximum number of spots expected
- Focal length of collimator
- Focal length of SH grid
- Optical parameters of the WFS system
- Type and list of Zernike polynomials to be fit in the SH analysis
- Orientation of the CCD frame
- Parameters for wave front computation
- Parameters for create an artificial SH grid for simulation and calibration

The SH control system will be able to perform single/multiple mirror frame analysis. In case of multiple analysis, each single frame will be analyzed separately and finally the average of the results will be automatically computed. The output of the SH analysis control software will furnish the following information:

- The optical parameters of the telescope used in the analysis
- The WFS parameters
- The description of the mirror and reference frame analyzed
- Date and time of computation
- The number of initial spots detected and the final number actually considered in the analysis
- The resulting Zernike coefficients with their internal error
- The resulting optical quality expressed in terms of percentage
- Some statistics derived from the correlation analysis
- Diagnostics in terms of movements of M2 needed in order to physically remove the computed aberrations.

A list of main control commands for SH control software system is the following:

- Switch on/off stepper motor driver
- Set position for 2 mm entrance hole positioned on the focal plane of the telescope (in order to find the star selected for SH frame in the field of the instrument)
- Set position for 8 mm entrance hole positioned on the focal plane (in order to take SH frame of the star)
- Switch on/off laser diode
- Set position for the laser diode+pinhole (required to take a reference frame)
- Set laser intensity to x value
- Initialize the system (default motor home position and laser diode off)
- Set pick-up mirror on optical path
- Set pick-up mirror in home position
- Set new SH flange position
- Move SH flange to next position
- Stop SH flange motion
- Read actual motor position
- Read actual laser diode intensity

6.3.3.4.10.4 Basic Active Optics Functions

For both M1 and M2 mirrors, the image quality is measured with a wave front sensor analyzer, the SH, placed on the telescope. The wave front sensor delivers centroid coordinates for all SH subapertures. The Image analyzer receives the centroids, calculates optical aberrations, and from these it calculates corrections to forces for the M1 supports, and corrections to M2 centering and focusing. More in detail, the active optics control software module will run on LCU#3 and will perform the following main actions:

- M1 active optics control software workflow: comparison between information coming from SH analysis, (actuators correct position configuration), and both actual temperature and telescope elevation axis position, (actuators actual position configuration). From these information it can be determined the corrections to be applied for the next actuators position configuration.
- M2 active optics control software workflow: comparison between information coming from SH analysis, (Hexapod correct legs position), and telescope elevation axis position, (Hexapod actual legs position), in order to obtain the corrections to be applied for the next Hexapod legs position configuration.

6.3.3.4.10.5 Operational modes

The possible operational modes for active optics control are essentially two: automatic mode and semi-automatic mode. For both, obviously, it is possible to apply differentiated or combined active control to M1 and M2.

1. Automatic mode: the corrections are calculated, stored and executed by the active optics functions.
2. Semi-automatic mode: the corrections are calculated, stored but not executed. This mode can be used in order to be able to control when corrections are done or not done. In this case it is left to observer the last choice to apply active corrections.

6.3.3.4.10.6 Control Functions

A preliminary list of main control commands foreseen for active optics on M1 and M2 is the following:

1. Startup active optics analysis system
2. Shutdown active optics analysis system
3. Set cyclic closed loop corrections
4. Set cyclic open loop settings
5. Set one open loop setting
6. Store the defocus value
7. Initialize M1
8. Initialize M2
9. Set number of measurements
10. Set number of image analysis to be averaged
11. Set delay between successive actions
12. Set targets for active optics operations: (none, M2 only, M1 and M2)
13. Set type for active optics operations: (absolute, differential)
14. Set integration time for reference
15. Set integration time for star
16. Plot of image analysis reference spot
17. Plot of image analysis star spot
18. Plot of image analysis residual error vectors
19. Close graphics
20. Set focus offset
21. Make image and evaluate optical quality without making any correction
22. Enable/disable optics control in open loop
23. Perform automatic image quality correction
24. Enable/disable regular automatic image quality correction

6.3.3.4.11 Optics Control

This section contains a preliminary description of main control functions needed to operate on two main mirrors and on some optics auxiliary devices, such as the ADC, intermediate lenses corrector group and filter wheel.

6.3.3.4.11.1 M1 Basic Functions

A preliminary list of main control commands foreseen for M1 is the following:

1. Start/stop M1 thermal control
2. Set M1 temperature reference
3. Set virtual fixed points positions
4. Set virtual fixed points to nominal values
5. Read virtual fixed points positions
6. Update nominal value table
7. Read nominal value table
8. Set address for active actuator n. y
9. Set position x for actuator n. y
10. Read position x for actuator n. y
11. Read charge cell n. z for radial supports

6.3.3.4.11.2 M2 Basic Functions

A preliminary list of main control commands foreseen for M2 is the following:

Power on/off hexapod motor drivers
Set velocity for hexapod legs movements
Get velocity for hexapod legs movements
Move hexapod legs to absolute position
Get hexapod legs movements status
Set hexapod pivot point
Give hexapod pivot point
Get hexapod legs position
Get hexapod error
Stop hexapod motion
Set M2 focus control to manual mode
Set M2 focus control to active optics automatic mode
Set M2 focus control to active optics semi-automatic mode
Set M2 focusing device to given absolute position
Move M2 focus given amount relative to current position
Set M2 focus position to latest correct value (for semi-automatic mode only)
Set M2 centering control to manual mode
Set M2 centering control to active optics automatic mode
Set M2 centering control to active optics automatic mode
Set M2 centering to center position
Set M2 centering to given absolute position
Move M2 centering given amount relative to current position
Set M2 centering position to latest correct value

6.3.3.4.11.3 ADC Basic Functions

A preliminary list of main control commands foreseen for ADC is the following:

1. Set ADC to position corresponding to requested RA/DEC at next calculated time
2. Set ADC to position corresponding to requested zenith distance
3. Set ADC to position corresponding to actual telescope position
4. Set ADC to requested absolute position
5. Set ADC to minimum separation
6. Set ADC to maximum separation
7. Relative change of ADC position
8. Power on/off motor drivers

6.3.3.4.11.4 Motorized Optical Auxiliary Devices Basic Functions

A preliminary list of main control commands foreseen for intermediate lenses devices is the following:

1. power on/off motor drives
2. move lenses corrector group in working position
3. move lenses corrector group in home position
4. read lenses corrector group motion status

6.3.3.5 Mode Switching

This section is intended to take care of all the possible states of telescope control system and its subsystems. The functions and tools used to make transitions between all working states are also described. More in detail, It will be described the group of functions foreseen for the following operations:

- Switch the telescope control system as a whole system between all acceptable operational states;
- Switch single subsystems between different states.

6.3.3.5.1 Subsystem States

As underlined before, the TCS is responsible of the functional coordination for many subsystems on the telescope. It is therefore important to underline all the possible functional relationships between telescope TCS and its subsystems. For application of the subsystems operational states to TCS, the following specifications can be made:

- A subsystem that is controlled by a TCS system in an **operating** state, and that is considerable as in the light path of the telescope, must be in **on-line** state; all other states are not permitted.
- A subsystem that is not controlled by a TCS system in an **operating** state can be in any state except **on-line**.
- The standard function "subsystem stand-alone" takes the subsystem from state **stand-by** to state **stand-alone**. For TCS subsystems, the function is only possible if the subsystem will not be in the light path of a telescope in an **operating** state. In other words, if the TCS is or is going in **operating** state, the standard function is not permitted.
- The standard function "subsystem on-line" takes the subsystem from state **stand-by** to state **on-line**. For TCS subsystems, the function is only possible if the subsystem will be in the light path of a telescope in an **operating** state.

6.3.3.5.2 LCU States

At each time the LCU is in one of the following operational states:

- **OFF** : in this case all the LCS functions are disabled. The LCU is not operational, in the sense that it can't give any reply. Of course the status OFF can't be returned from the LCU.
- **LOADED** : in this state all application software is loaded, the database is configured, the software is initialized and ready to initialize the hardware.
- **STAND-BY** : during this state all the LCU software is loaded initialized and operating, all hardware is initialized. But the devices are in STAND-BY mode, e.g. possibly switched off, brakes engaged; the actual state of a device in STAND-BY mode has to be defined for each subsystem.
- **STAND-ALONE** : in this state all software is loaded and initialized, all hardware is initialized, power is switched on (this is the state normally used for maintenance and for off-line instrument operations). The difference with ON-LINE mode is that the communication between LCUs is disabled.
- **ON-LINE**: this is the state for normal operative LCU functions. All software is loaded and initialized, all hardware is initialized and the power is switched on. All devices are operative.

6.3.3.5.3 *TCS States*

At each time the TCS is in one of the following operational states:

- **OFF** : in this case all the TCS functions are disabled. Normally it is only permitted to use the User Interface in disable or simulation mode.
- **STAND-BY** : this state can be addressed as the normal day-time state. All the transitions from OFF mode to STAND-BY mode is done with a so-called "cold start" procedure, that involves the switching active all the main subsystems. It must be remarked that all the subsystems involved in the cold start must not be in on-line state. During the STAND-BY state, some TCS functions are available:

  1. Reading and updating of site monitor data;
  2. Supervision of rack cooling system;
  3. Supervision and diagnostic of HBS pressure system;
  4. Reading and updating of some telescope subsystems telemetry;
  5. Supervision of enclosure air conditioning system;
  6. Communication with the building management system;

- **OPERATING** : This is a particular operational state that is normally splitted in four sub-states, defined below. The transitions to and from the OPERATING state are done by functions that will be fully described in the following sections. The four sub-states can be disturbed by different kinds of effects, which can affect their normal operations. From the Mode switching point of view these four sub-states are not considered as separate states. The four states included in the term OPERATING are:

  1. **IDLE**
  2. **POINTING**
  3. **TRACKING**
  4. **PAUSED**

  In these states all the telescope subsystems involved are in ON-LINE state, and controlled by TCS. Other subsystems can be in any other state, except ON-LINE. Of course, all the coordinating functions needed during observations must be active and ready for operation, (POINTING and TRACKING states). The telescope is under position control loop unless it is in any interlock event o switched to PAUSED state. Infact, during states IDLE and PAUSED the telescope is not moving, with the main difference that, during IDLE operational mode the position loops are closed and brakes not engaged, while in PAUSED mode the position loops are open and brakes engaged.

6.3.3.5.4 Functions for Mode Switching

In this section all the higher level functions needed to switching between the operational modes mentioned above will be described. Here it will be considered mode switching for the telescope at a whole system level. These functions can be summarized in the following considerations, (see Fig. 5.8):

- TCS COLD-START: this function takes the TCS from state OFF to state STAND-BY
- START TELESCOPE: this function takes the TCS from state STAND-BY to IDLE. This means the following operations:

  1. Starting all telescope subsystems necessary for observation;
  2. if any of these subsystems is in STAND-ALONE state, it must be first switched to state STAND-BY;
  3. all subsystems are started, i.e. taken to state ON-LINE. The telescope is set under position control in order to be positioned close to the actual position;
  4. When the start-up is ready, the telescope is always under position control, but standing still, ready to move immediately after next pointing command;

- POINTING: These functions, switching the TCS to state POINTING from any other OPERATING state, are the normal operations, already described in the pointing section of this document. The state POINTING is a transient state; when a pointing is ready, there is automatically a switch to IDLE or TRACKING states, depending on if the pointing was a technical or astronomical one. Namely, the technical pointing is involved when the telescope must be pointed to AltAz coordinates, while the astronomical pointing concern with all other types.
- PAUSE: this function takes the TCS to PAUSED state from any of the other OPERATING modes. Its role is to stop the telescope and to switch it to a safe status at a requested home position. This means that brakes are activated and control loops deactivated. Most functions are left switched on, in order to furnish the quick switch back to one of OPERATING state. So, the subsystems controlled by TCS remain in ON-LINE state.
- TELESCOPE CLOSE: this function switches the TCS from any of the OPERATING modes to state STAND-BY. The telescope is moved to its predefined home position for all axes, then all control loops are switched off, all subsystems and TCS functions shut down to state STAND-BY. This is the normal "end of observing night" procedure.
- TELESCOPE SHUT DOWN: of course, this function takes the TCS to OFF state.
- SET INSTRUMENT PARAMETERS: these functions are useful in order to pass all instrument parameters to TCS core. The TCS, infact, needs some pointing and autoguiding information related to instrument control, such as detector rotation offset, (i.e. the angular difference between detector up-down line and rotator zero position), pixel size of detector, pointing axis offset, (i.e. the position offset from center of rotation of the wanted reference point).

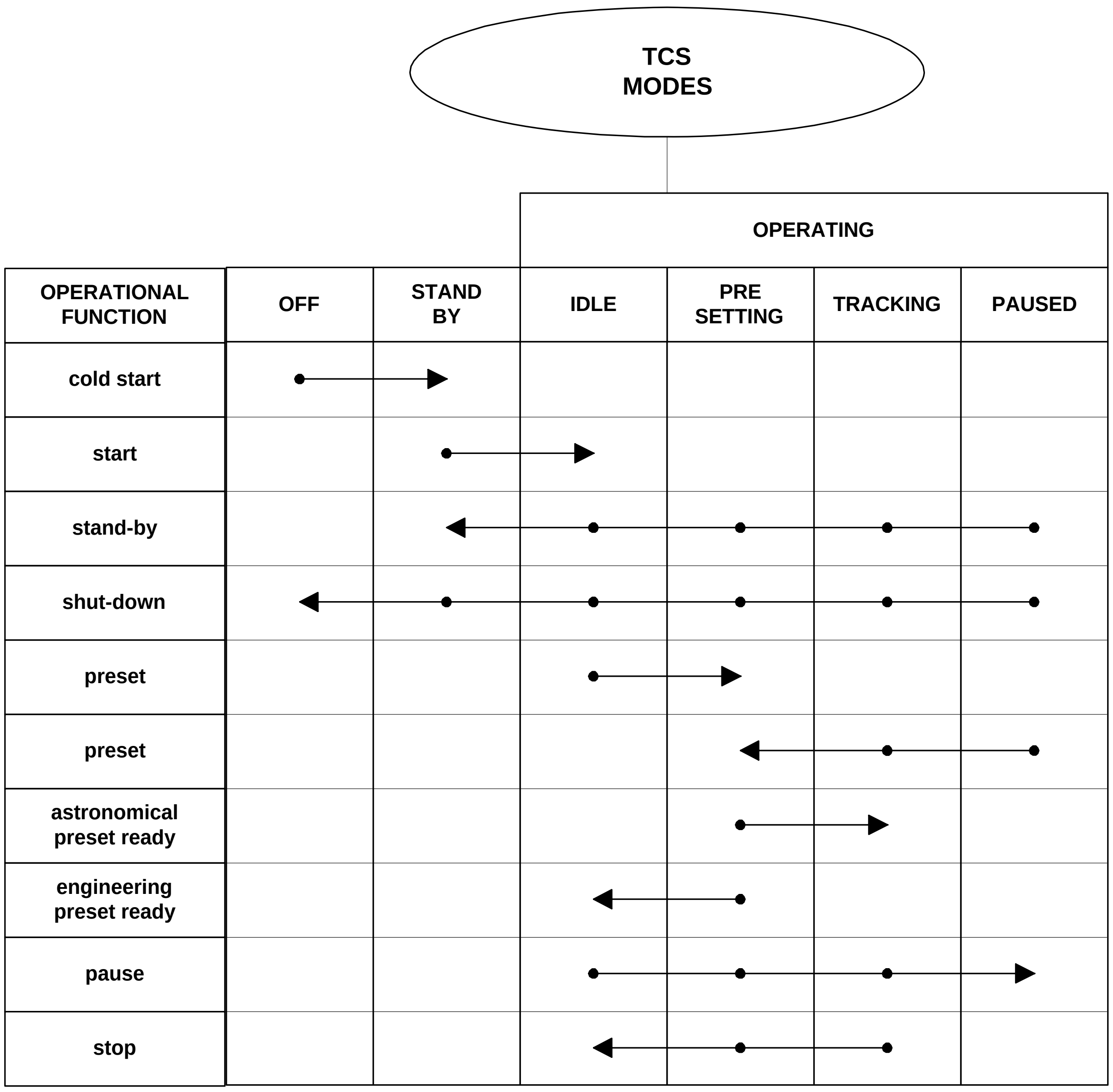


Fig. 5.8 - TCS Higher Level Functions for Mode Switching Transitions

There are other functions, considered at a lower level, used to control the mode switching between different subsystems, at LCU level. These functions can be summarized in the following considerations, (see Fig. 5.9):

- COLD START: the LCU is performing a cold start. This operation takes the LCU to the STAND-BY, STAND-ALONE or ON-LINE states.
- WARM START: this operation function takes LCU to the STAND-BY, STAND-ALONE or ON-LINE states. The difference with the cold start is that in this case the software initialization and loading was already done.
- INITIALIZING: during this operation the LCU is performing essentially the hardware initialization. It takes the LCU into the STAND-BY state; this operation would be considered as a sub-function of COLD and WARM START operations.

- SHUTTING DOWN: it takes the LCU in LOADED state, causing a process of switch off for all devices and a shut-down for all processes of the LCU.
- SIMULATION MODE: under simulation the direct connection of LCU with all physical devices is disabled. This situation can be reached by the LCU only if its starting mode is one of LOADED, STAND-BY, STAND-ALONE or ON-LINE states. Exiting from simulation will take the LCU into the LOADED state.

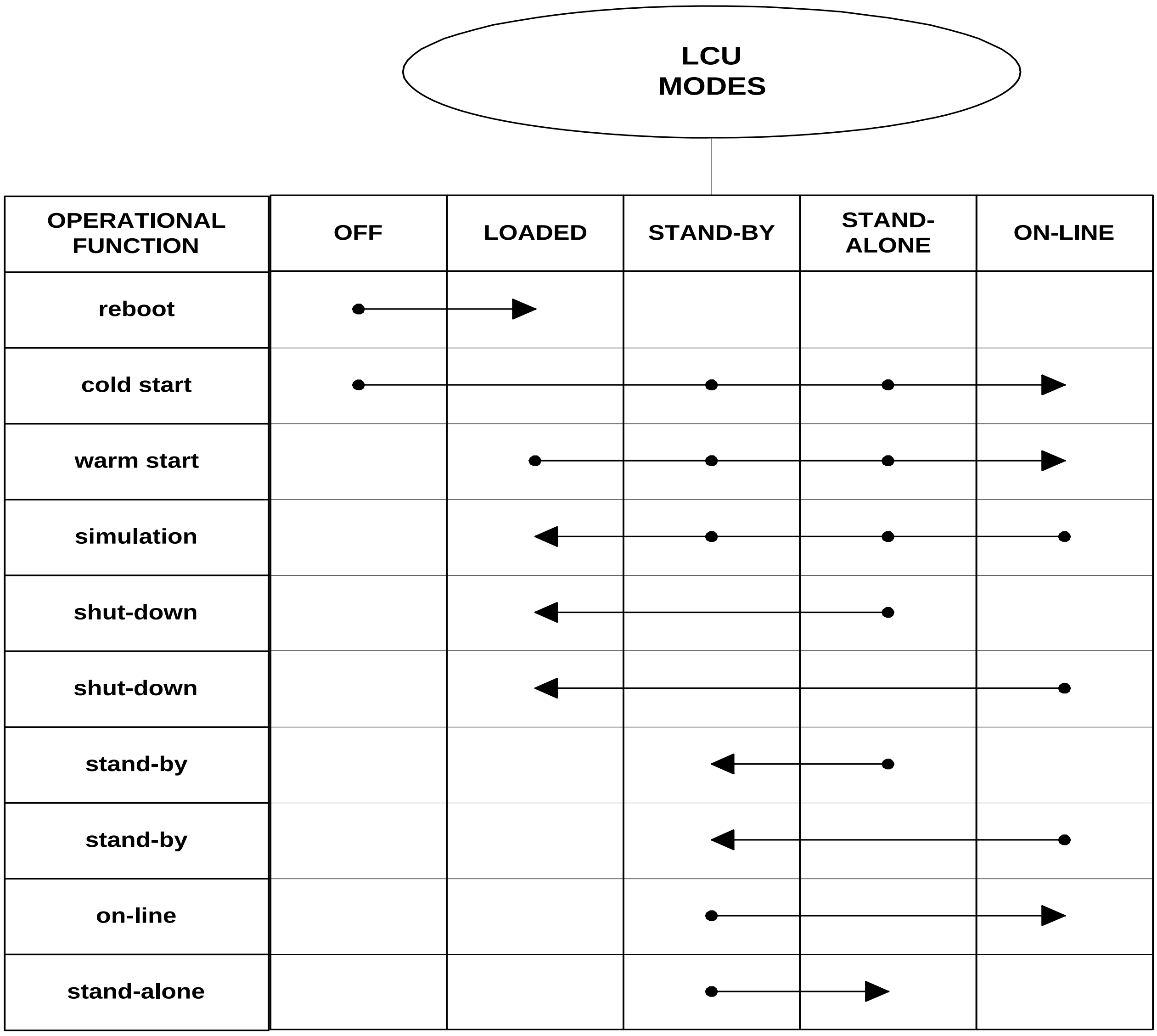


Fig. 5.9 - LCU Switching Functions for Mode Switching Transitions

6.3.3.6 Engineering and Maintenance

6.3.3.6.1 General Requirements

For VST engineering and maintenance purposes, the complete functionality of all LCUs, all their subsystems and all high level functions shall be available. This implies, in particular, the development of a well defined engineering user interface, useful during low-level tests, calibration and maintenance of the telescope control software. At a low level, all I/O signals must be individually accessible for reading or writing and all moveable and adjustable devices can be individually manipulated. Furthermore, most VST LCUs will have other low level functions that are normally used by high level functions in a way completely invisible for an observing user, but that are directly usable for engineering and maintenance purposes. Available and accessible mean that special Engineering and Maintenance User Interface panels and status forms will be created, using the capabilities of the User Interface system. Also other standard functions in the LCU and on higher levels, like alarms, error logging, logging, event handling etc., are partially designed for engineering and maintenance purposes.

6.3.3.6.2 Control Loop Tuning

For engineering purposes, i.e. test and trouble shooting, The VST control software will be able to enable/disable functional micro steps, which are not individually available via the normal command and user interface, but are part of a standard functional sequence. Examples of such micro steps can be: pointing model calculation during tracking, refraction calculation, mirrors positioning. But, for the control loop a set of special functions will be available to assist in the engineering job of tuning a cascaded control loop, for the axis where a combined hardware/software solution of torque/velocity and position control is implemented. Here it follows a preliminary list of these functions:

- Perform a particular position step;
- Move axis with a prefixed/calculated acceleration;
- Move axis with a prefixed/calculated velocity;
- Make particular calculated movement, with specified parameters, measure the input/output signal ratio and store measured gain/frequency and phase/frequency data to be used for test plots.

6.3.3.6.3 Engineering User Interface

As for ESO LCU Common Software, for the VST control software will be used a set of tools, provided partially by VxWorks an partially by LCS, such as:

- Low level debugger;
- Utilities to inspect the status of tasks running on the LCU;
- Utilities to measure the execution time of routines;
- Utilities to check the load of the CPU;
- A source code debugger;
- Utility to access to all parameters in the local database;
- Utility to access to all I/O signals;
- Possibility to log and sample signals and items in the local database.

Furthermore, it is foreseen the inclusion in the VST Telescope Control Software of a well-defined Engineering User Interface. Such EUI is able to use and test the LCU in stand-alone mode. It will operates at two different levels:

- LCU operating system shell (TORNADO VxWorks)
- Unix workstation EUI

In order to activate all the requested capabilities, the EUI must have the following features:

- The EUI will use the Unix WS utility to send a command to any process on any node and wait for a reply.
- The EUI will accept a file with a list of commands to be sent to the specified destination.
- The EUI will use the Message System to send the commands to the LCU under control.
- The EUI will also use a file defining error and alarm messages to print error messages in the command window and to display alarms.
- The EUI will also be able to display help text on commands. For the selected process, the EUI will list all commands and for each command list the parameters with their type and range, indicating which parameters are optional and also display a text describing the command.
- The EUI will log all messages sent and received into a file. An utility can extract information from the file according to some basic filtering criteria like destination node, destination process, command, reply, error message, log message etc.
- The EUI will also be able to act as a log server. It will be able to receive log messages from the LCU and display them on the screen.
- The EUI will have an utility to display a defined set of database parameters. It will also be possible to update the values of these parameters.
- Finally the EUI will also be able to monitor a given set of parameters, i.e. to display them with a given update period.

6.3.3.6.4 Maintenance

The VST maintenance philosophy is based on the fact that all signals and local database parameters will be accessible locally and from other nodes. This doesn't only include signals directly connected to the LCU. It must be also possible to access signals connected to systems below the LCU. All moving and adjustable parts have to be individually movable and adjustable from another node. Similarly it must be possible to adjust configuration parameters for motors and to access the status of all movable parts, e.g. encoder position, temperature, current etc. The event monitoring can be used to monitor the behaviour of the LCU system. Preventive maintenance has to be carried out periodically for each subsystem. This will be defined in a file containing the list of actions to perform, frequency, urgency of action etc.

6.3.4 Astronomical Computations

In order to standardize the astronomical computations, which are quite CPU intensive and complex, there shall be standard functions made available which provide the needed values in observation coordinates out of the presetting values provided by TCS. Following the ESO implementation requirements, all the astronomical coordinates will be delivered at the same time to all the axis control LCUs. Then, each LCU will use those needed for its own calculations. In the case of the three main axes of the VST, this calculation module operates on the celestial object coordinates, as provided by the last presetting command. After handling of the various offsets, explicit offsets, proper motion offsets, velocity offsets, autoguider offsets. It derives the theoretical axis position to be used by the position servo loop through the following steps:

1. starting parameters, alpha, delta, hour angle, latitude.
2. general coordinate transformations applicable to all rotating axes.
3. AltAz
4. Refraction correction
5. AltAz, parallactic angle
6. Pointing telescope specific transformations
7. ADC device specific transformations (optional)
8. AltAz, parallactic angle for trajectory shaping calculations.

6.3.5 Time Reference System

The time reference system that supplies a reference time for astronomical calculations and CPUs synchronization will be based on the use of a Global Positioning System (GPS) receiver. The LCUs receive the UTC timing information via fibers connected to ESO standard VME TIM (Time Interface Modules) boards. The LCUs require accurate timing to perform star tracking commands, so they are connected to a dedicated time bus whose accuracy on the hardware level is about 10 microseconds. The other system processor, i.e. the WS, doesn't require a so accurate synchronization, resulting in a slightly lower accuracy. VST telescope will use the facilities existing in ESO observatory; OAC will provide VME LCUs equipped with TIM boards, that will be connected to the GPS facilities provided by ESO. The TIM is a printed-circuit board with a connection to the TRS time bus and inserted in a VMEbus slot of a LCU. A TIM device consists of three main parts:

- The UTC part
- The timer part
- The interrupter

From the software point of view, the TIM device will be used by LCU application programs through direct calling a standard TORNADO device driver. The layout of the time reference system provided for the VST is shown in Fig. 5.10.

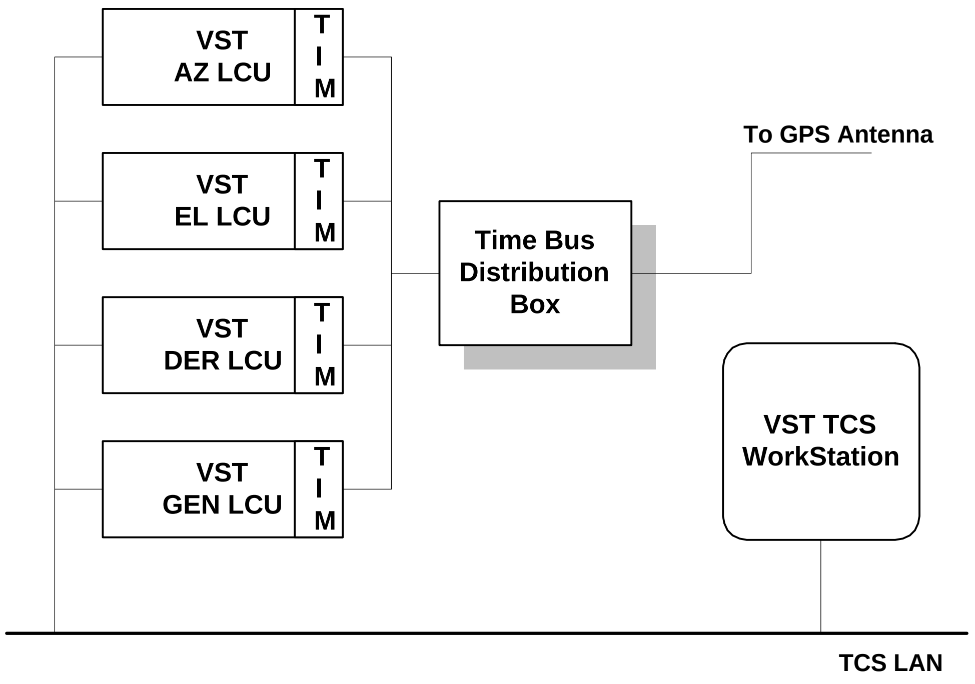


Fig. 5.10 - The VST Time reference System Design

The tim driver is accessible through the TORNADO I/O system interface, i.e. the open, close and ioctl system calls. To get access to the device a user has to open a communication channel to it by calling the open routine. This system call returns a file handle with which the channel is identified in any further driver calls. The opened channel can be closed by the close system call. All other functions provided by the tim driver are invoked by the ioctl system call, with its parameter specifying the action requested to be performed by the driver.

6.3.6 *Telescope software interfaces*

6.3.6.1 Workstation - LCU Interface

The VST TCS control and data flows are designed following the main ESO VLT Common Software guidelines. To reflect this point, the strategy adopted for the information exchange between telescope WS and LCUs will be based on the following top level data flow diagram, (Fig. 5.11). From the workstation host, during initialization phase, the LCUs boot TORNADO remote kernel. After executing own bootscript, the LCUs perform the downloading of LCC module, including drivers and database. During this phase all devices and I/O lines will be loaded and configured. When a command is issued from the WS, the Command Interpreter performs a syntax check, using a prefixed command definition table. If the command syntax is correct, the Command Interpreter will use another prefixed table to interpret the command and to execute it. Then the Execute Command module will access to the local database to update the system data involved in the previous task executed. Finally, through the scan system, the WS database will be updated and the reply of task completion will be returned to WS application.

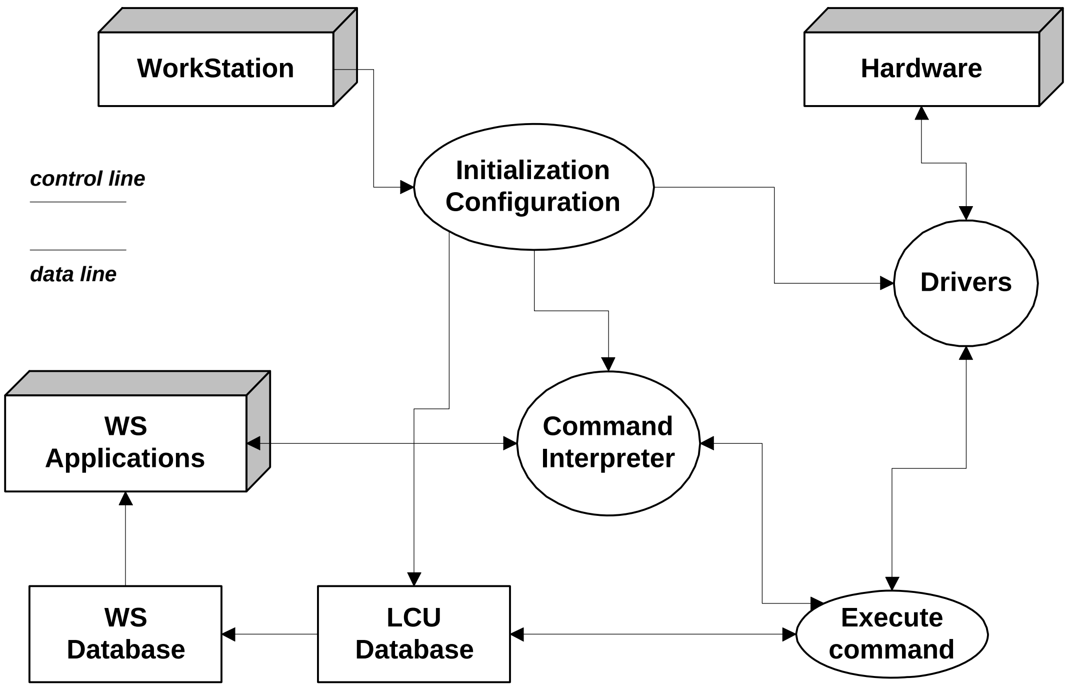


Fig. 5.11 - WS-LCU Data Control Flow Diagram

6.3.6.2 Human Interface

The main human interface foreseen for the VST software will be the GUI and EUI. These functionalities will be used to configure a suitable set of panels, forms, menus, etc., which will allow user friendly operations, for the different kinds of users, of all TCS functionality. In principle, all functions are available via the user interfaces. However, many functions will only be available on panels/forms specially designed for engineering and maintenance. Since the user interfaces, together with its tools, is a flexible, easy to change, configurable system, which mainly operates on data stored in the on-line database. The exact amount and layout of TCS panels and forms will be defined after the PDR phase.

6.3.7 Database Handling

The hierarchical organization of the VST TCS database allow mapping of the physical objects constituting the different hardware components into database structures able to describe the logical relations existing between these elements. The VST database design and development will follow as much as possible the ESO VLT Common Software database organization. Since in the VST also most of the applications will be split between WS and LCUs, the database branches supporting them will also be distributed on both platforms. Nevertheless, they can be described in a single file. This feature, allowing a global and synthetic view of the database helps in the design phase and reduces the risk of errors during the implementation and maintenance phases. The VST database will have a modular structure used for code development and data structures definition. Each individual module will describe its own database branch. This branch would have a root point with name equal to the module name, the alias being the same, typed in lower case. All applications will refer to data in this branch following these rules:

- New classes specific to the module will be defined in a file and stored in a dedicated dbl directory of the module.
- The module database branch will be described by declaring the instances of the related classes in the appropriate dbl files.
- The same database branch will be used to build test environments.
- During integration, the dbl files from the different modules will be installed and the global database created by attaching the module branches under the same root point.

An effort will be done in order to use as much as possible the hierarchical structure of the database, in terms of its correspondence to the hierarchical structure of the software. For example, if a motor has a tachometer, it should be correct to make the point representing the tachometer as a child of the point representing the motor.

6.3.8 System Logs

Using standard CCS mechanisms, all major events in the system are logged. Examples are: malfunctions, alarms etc. Logging is a standard mechanism for each LCU of the system; however each LCU subsystem logs its own particular events, depending on the functions operated by the unit. Examples of telescope axis control logging could be, for instance, limit switch activation, failure in power amplifiers. The logging system of the VST will follow the baselines used for VLT Common Software, including log message format. Following this idea, the VST logging system will support automatic logging of events, in particular, messages received by or sent from a process, local database write/read accesses, change of state of digital input/output signals, change of analog signals. Of course the user of the logging system can disable or enable the logging of these events. The VST LCU software will use logging to log its activity (automatic and error logging) using specific formats. A preliminary design requirements for this format is the following:

- The definition of a log will be stored in the logging database, where there is specified the number of fields, their position, name and type.
- In addition, some common fields will be defined for each log:
  1. System name
  2. Log format number
  3. Originating environment
  4. Originating process
  5. Time reference

In order to perform logging operations, some functions will be provided. A preliminary set of these functions is the following:

- LogText(): used for logging of a user defined log text
- EnableLog(): it enables the logging system through the message system
- DisableLog(): it disables the logging system
- EnableLogDBL(): it enables the logging of WS database
- EnableLogDBR(): it enables the logging of LCU database
- DisableLogDBL(): it disables the logging of WS database
- DisableLogDBR(): it disables the logging of LCU database
- EnableLogDIO(): it enables the logging of Digital I/O signals
- EnableLogAIO(): it enables the logging of Analog I/O signals
- DisableLogDIO(): it disables the logging of Digital I/O signals
- DisableLogAIO(): it disables the logging of Analog I/O signals
- StartLog(): it starts all logging operations enabled
- StopLog(): it stops all logging operations enabled
- GetLogEv(): it will return a list of all logging operations enabled
- SetLogRate(): it will set the minimum logging time frequency. That's in order to reduce the LAN traffic.

6.3.9 Priorities

Different priorities are assigned to the tasks performed by software. Of course the time critical functions are related to the servo loops. They must be absolutely executed with maximum accuracy in terms of synchronization and performance. The LCU Command Interpreter, discussed above, is dedicated to all standard commands concerning the axis, and can communicate with time critical functions. Low priority tasks are used to monitor all the data needed to have an image of the system status in the database. The VST priorities design will follow the above requirements.

6.3.10 HBS Software Interface

The TCS will be responsible for the control of the Hydrostatic Bearing System for VST azimuth axis movement. The main role of TCS is of course to provide a on-line diagnostic on HBS system status. A preliminary list of I/O data available for TCS is the following:

- Oil pressure.
- Oil temperature at the level of pads.
- Oil flow.
- Control of the valve status.
- Control of the hydraulic subsystem.
- System switch on/off.
- System fault.
- Interlock monitoring.

In case of any fault, an interlock signal will occur. The axis will be stopped and an alarm will be reported to TCS.

6.3.11 *Temperature Monitoring Software Interface*

During normal operations, the TCS telemetry data monitoring includes also a set of temperature sensor data, located around the telescope in well-defined positions. These data will be monitored by TCS through dedicated electronic devices, specialized for thermal conditions handling, directly connected via serial link on the related LCU. The software used will be ESO-compliant. Some of these temperature data are referred to motors, inlet control rack, outlet control rack, tube assembly, top ring, HBS and other positions on the telescope.

6.3.12 *Telescope Control Software Test, Simulation and Interfaces to the DFS*

6.3.12.1 Test Requirements

The LCU has self-test software which runs automatically at start-up and at other times on request. It checks that the boards configured with the LCU are present and configured properly. This is done for all I/O devices that have registered to the LCC including also CPU, time interface etc. Then it sends a self-test command to the application programs of all devices, listed in a configuration table in the database. At this level the self-test will not move any mechanical device. Testing of the devices will be performed in a sequential way.
To facilitate test and maintenance at the LCU level, test and maintenance software will be provided. Three levels of test and maintenance software will be supplied:

1. **Monitor level**: the LCU will have a monitor program, running whenever it is operational as a background task, to continually carry out checks such as power supply levels, temperatures, pressures, hardware status signals, (limit switches, power amplifier errors, encoder errors, emergency switches etc.), levels of analog signals, correct termination of executed commands (timeouts) etc. An error report will be sent to the control node on any detected irregularity.
2. **Self-test level**: the application program for each device on the LCU will accept a self-test command. It will be sent to the device at start-up and on request. The self-test will be available at different levels:
   - Checking hardware configuration and initialization procedures; it will check that the correct configuration of boards with the correct setup is present;
   - Perform a complete test of the electronic modules used. The purpose of this test is to check if the module is working correctly. The test will only cover one module at a time. It will be a low-level test, using as much as possible existing drivers, but if necessary also directly access registers on the board. The test will be automatic, giving the positive results or indicating a failure. The module will be tested without disconnecting any cables and without connecting any test equipment. This will make it possible to run the test remotely, e.g. from the telescope control room. The self-test will restore the initial status, after terminating, if the initial status was not an error status, otherwise, the system will be switched into the STAND-BY status.
3. **Simulation level**: see par. 6.3.12.2 below.

6.3.12.2 Simulation

Simulation of the VST Telescope Control Software is available at different levels. At the LCU level where all LCU hardware is simulated and at device level where each device can be simulated individually. This means that application software for each device will accept the commands to enable/disable simulation mode. The remote database access and the time reference system are not included in the general simulation, but have their own simulation mode. **The LCU software is not allowed to enter simulation mode if any hardware device is active** (for example a motor that is moving). In simulation mode the LCU will be able to work without external devices. Hardware input signals will return default values set up at configuration time. It will be possible to change these values dynamically. At simulation level the application program for each device on the LCU will accept the command "enable/disable simulation". In simulation mode it will simulate all hardware controlled by the device. The software will respond to commands as in normal operation but no hardware will be activated. Hardware status will return default values set up at the configuration of each signal. Applications in simulation mode will return reasonable values reflecting the state of the devices it is controlling and the commands previously received. Commands which normally take some time to execute, if in simulation mode, return without delay. When a device or the whole LCU returns to normal operation, the software of that device will have to be reinitialized. The device will get the status non-initialized when exiting from simulation mode.

6.3.12.3 Interface to the DFS

The operation of VST telescope will be compatible with the concept of Data Flow System (DFS) developed for VLT by ESO Data Management Division. The DFS handles the flow of scientific data associated with the operation of the telescope. Its main functions are:

- Preparation and processing of observing proposals
- Scheduling of observations
- Execution of observations
- Reduction of data
- Quality control of data
- Archiving of information

The VST TCS will be interfaced to the DFS as specified above.

6.3.13 VST Software Safety Planning

6.3.13.1 Preliminary Hazard Analysis

The purpose of this task is to perform and document a PHA to identify software safety critical areas, evaluate hazards and identify the safety design criteria to be used. In this section we will provide a preliminary list of safety provisions and alternatives needed to eliminate hazards or reduce their associated risk to a level acceptable to ESO. The VST software PHA will take into account the following items in order to identify and evaluate the possible hazards:

- Operating, testing, maintenance and emergency procedures.
- Software hazardous components, (e.g. WS I/O devices handling, WS file system maintenance, software tools installation and use, WS and LCU TCS access, etc.).
- Safety related interface considerations among various elements of the system, e.g. hardware and software controls, safety critical software commands and responses. Inadvertent command, failure to command, untimely command or responses, designated undesired events will be identified and appropriate actions taken into account to incorporate them in the software specifications.
- Software fail-safe design considerations.

6.3.13.1.1 LCU Failure Mode Operation

This section describes a preliminary set of operational functions foreseen for VST software reactions to errors and abnormal situations. It also define maintenance software, which can be used to reduce the probability of errors. The basis of error handling is that all error conditions are checked and that all errors and abnormal events are reported to the error handling node. Error and abnormal events are subdivided into the following categories:

- FATAL: no recovery is possible. The LCU will shut down all hardware activities in a controlled way and abort all software activities. The hardware will also be able to shut down the system in case of CPU failure.
- SERIOUS: In this case the LCU cannot perform what it was supposed to do, but there is no need to shut down. Devices causing or affected by the error will be stopped.
- WARNING: An anomalous situation is reported, but the LCU can continue to work.

In most situations the errors and abnormal events can be grouped and assigned to any of the above categories. For error/failure reporting and recovery, the VST LCU control software will take into account the following considerations:

1. Associated with the errors there will be a file containing the error message corresponding to each error. The error message has the main role to communicate a clear, unambiguous and unique description of the error. The error message will also be detailed enough to make it clear what caused the error itself.
2. With the abnormal events, there will be a file containing the alarm message corresponding to each abnormal event. The alarm messages will give a clear, unambiguous and unique description of the abnormal event. The alarm message will also be detailed enough to make it clear what caused the abnormal event.
3. Alarms will be handled differently during the engineering phase and when the LCU subsystem is integrated into the VST software:
   - During the engineering phase, i.e. before the integration between LCS and CCS, the alarms will be sent to the EUI, using abnormal event handling;
   - After the integration between LCS and CCS, the alarms will be handled by the CCS on-line database. To enable this, the LCU has to be configured to report all events on signals and local database parameters to the CCS on-line database for all signals and parameters which will possibly trigger an alarm. The LCU configuration setup will include the configuration of the event monitor. Delivered with the subsystem will also be a list with all possible alarms from the LCU.
4. It will be possible to trigger any error or alarm message from any application from software, having the possibility to simulate any error or alarm condition interactively.
5. The LCU subsystem will try to recover from failures. Retries will be done when appropriate on failing devices. If it is not possible to recover, it will shut down the failing device, parts of the subsystem, or the whole subsystem.
6. Retries must be also included in the local command handling so that recovery of the most common time-out and no-answer situations is attempted, before coming up with an error.
7. The warnings produced at every trial will be logged, to be able to measure reliability.

In terms of failure mode performance a brief description must be defined for every LCU subsystem in the corresponding functional specification:

- Normally there is no operation possible after a Fatal error, apart from shut down and restarting.
- In case of serious errors, various degraded operation modes might be possible, after unsuccessful recovery attempt.
- Normally there will be no warnings during normal operation. An high number of warnings when no hardware failure is involved, can only indicate a mismatch or marginal behaviour of the software and indicate the need to adjust it.
- It should also be avoided that working parts of the system are affected by failing units, for example because errors or abnormal events keep being reported and flood the system.

6.3.13.1.2 LCU Safety Control

Many LCU subsystems will be connected to one or more hardware stops. For the VST system we'll provide the following hardware stops:

- Emergency stop: a global stop and power-off of everything on and within the VST enclosure. This stop is activated via special push-buttons. In this case no action is possible by the LCU software, because the LCUs are switched off.
- Motion stop: it will stop a number of predefined axes, e.g. altitude, azimuth, dome rotation, etc. by hardware; this can be done by switching off power amplifiers and engaging motor brakes. All motion stop buttons in a given area will have identical action. An area in this sense is a physically separated working area, such as, for example, the telescope area. This system can be used during installation and maintenance to prevent accidents. Motion stops can be activated by special push-buttons or key switches. At software level, a LCU controlling a device connected to a motion stop will be informed of the event via a status signal. When a motion stop is activated, the LCU application software will shut down this device. This is to prevent the device from moving when the motion stop is deactivated, and before the subsystem has been restarted.
- Local disable switch: it disables the motion of a specific unit, e.g. the altitude motion. This can be implemented, for example, by switching off a power amplifier, or by disconnecting the unit from its power supply, or in some other way. The switch can only be manually operated. If there is a need for computer controlled power switching, there has to be an additional switch to accommodate this. An LCU controlling a device connected to a local disable switch will be informed of this stop via a status signal. When a local disable switch is activated, the LCU application software will shut-down this device. This is done to prevent the device from moving when the local disable switch is deactivated and before the subsystem has been restarted.
- Local power switch: it switches off power to a connected unit; a rack, an LCU or whatever. The action of the LCU software depends on which unit has been switched off.
- Interlock: it implies an automatic and immediate stop of an individual axis in case of a dangerous situation. The signal triggering the interlock action is a hardware signal and the action is completely performed by hardware. There are two groups of interlocks:
  1. Interlocks local to a subsystem, e.g. azimuth axis stop if bearing oil pressure is too low.
  2. Interlocks between two subsystems, e.g. manual control switch for altitude axis.

From the software point of view, an LCU controlling a device connected to an interlock will be informed of this stop via a status signal. When an interlock is activated, the LCU application software will shut down this device. This is done to prevent the device from moving when the interlock is deactivated, and before the subsystem has been restarted.

6.3.13.2 SubSystem Hazard Analysis

The purpose of this task is to perform and document a SSHA to identify hazards associated with the design of the instrument software, including component failure modes, critical human error inputs, hazards resulting from functional relationships between software components comprising each instrument, software performance degradation, functional failure. In particular, the analysis will take into account:

- potential contribution of software events, faults and occurrences (such as improper timing) on the safety of the subsystem.
- That the safety design criteria in the software specifications have been satisfied.
- That the method of implementation of software design requirements and corrective actions has not decreased the safety of the subsystem nor has it introduced any new hazards.

For any LCU subsystem, the following principles will apply:

1. For each subsystem there will be a description of all safety events and the corresponding actions to be taken by software.
2. Any subsystem software has to take its components to a safe state in any error, emergency or dangerous situation.
3. All subsystems have to define:
   - Their own internal interlocks;
   - Interlock signals that are needed in other subsystems;
   - Interlock signals needed from other subsystems;
4. The subsystem software will continuously monitor the state of its components (via input signals and other parameters) and if the components come into a state potentially dangerous, it will immediately stop the components. The monitoring of signals and parameters will be done by using the event monitoring and abnormal event handling before described.
5. The LCU software will always be able to accept a stop or abort command and execute them to stop ongoing activity within a prefixed time.
6. The LCUs will report all stops and interlocks as abnormal events.